T.C.
MİLLÎ EĞİTİM BAKANLIĞI

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

MİKRODENETLEYİCİ İLE LCD VE SERVO MOTOR KONTROLÜ

ANKARA 2014

Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.

Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.

PARA İLE SATILMAZ.

İÇİNDEKİLER

İÇİNDEKİLER i
AÇIKLAMALAR ii
GİRİŞ 1
ÖĞRENME FAALİYETİ–1 3
1. PIC16F877 İLE LCD KONTROLÜ 3
1.1. PIC16F877 Mikrodenetleyici 3
1.1.1. PIC16F877 Mikrodenetleyicinin Özellikleri 3
1.1.2. PIC16F877 Mikrodenetleyicinin Fiziksel İç ve Dış Yapısı 5
1.2. Lcd Yapısı ve Programlanması 17
1.2.1. LCD'lerin Genel Özellikleri ve Çeşitleri 17
1.2.2. Paralel Karakter LCD Yapısı 18
1.3. LCD Programlanması 32
1.3.1. PIC C ile LCD Kodu Yazımı ve Proteus Isıs ile Simülasyonu 32
1.3.2. Paralel LCD Uygulamaları 46
UYGULAMA FAALİYETİ 70
ÖLÇME VE DEĞERLENDİRME 72
ÖĞRENME FAALİYETİ-2 74
2. MİKRODENETLEYİCİ İLE SERVO MOTOR KONTROLÜ 74
2.1. Servo Motorun Yapısı 74
2.2. Servo Motorun Mikrodenetleyici ile Kontrolü 78
2.2.1. Servo Motorun Yazılım ile Kontrolü 80
2.2.2. Servo Motorun Donanım PWM ile Kontrolü 85
UYGULAMA FAALİYETİ 99
ÖLÇME VE DEĞERLENDİRME 100
MODÜL DEĞERLENDİRME 102
CEVAP ANAHTARLARI 105
KAYNAKÇA 112

AÇIKLAMALAR

ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Endüstriyel Kontrol Teknisyenliği**
MODÜLÜN ADI	**Mikrodenetleyici İle Lcd ve Servo Motor Kontrolu**
MODÜLÜN TANIMI	Bu modül ileri seviyede mikrodenetleyici uygulamalarınıtanıtan ve öğrencilerin üst düzey mikrodenetleyici sistemtasarımlarını geliştirmelerine yönelik bilgi ve becerilerin verildiği öğrenim materyalidir.
SÜRE	40/32
ÖN KOŞUL	523EO0020 numaralı Mikrodenetleyici Programlama Temelleri modülünü almış olmak
YETERLİK	Mikrodenetleyici ile display ünitesi yapmak ve Mikrodenetleyici ile servo motor kontrol ünitesi yapmak
MODÜLÜN AMACI	**Genel Amaç** Mikrodenetleyici ile ileri seviye uygulamalarını hatasız olarak yapabileceksiniz. **Amaçlar** 1. Mikrodenetleyici ile LCD kontrolünü hatasız olarak yapabileceksiniz. 2. Mikrodenetleyici ile servo motor kontrolünü hatasız olarak yapabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Elektronik Laboratuvarı **Donanım:** Mikrodenetleyici, mikrodenetleyici programlama kartları, mikrodenetleyici uygulama devreleri, elektronik elemanlar, güç kaynağı, osilaskop, sinyal jeneratörü, RF Servo motor, LCD, bread bord, el takımları
ÖLÇME VE DEĞERLENDİRME	Modülün içinde yer alan her öğrenme faaliyetinden sonra, verilen ölçme araçlarıyla, kendinizi değerlendireceksiniz. Öğretmen, modül sonunda size ölçme teknikleri uygulayarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek değerlendirebilecektir.

GİRİŞ

Sevgili Öğrenci,

Mikrodenetleyici İle Lcd ve Servo Motor Kontrolü Modülü ile LCD kontrolü, RF Servo motor kontrolü ile ilgili temel yeterlikleri kazanacaksınız. Kontrol sisteminde, daha önce öğrendiğiniz PIC16F84 mikrodenetleyicisinden çok daha fazla özellikleri bulunan PIC16F877'yi kullanacak ve kontrol programlarını da PIC-C dili ile gerçekleştireceksiniz.

Böylece PIC ailesi içerisinde farklı bir mikrodenetleyiciyi ileri seviyede bir programlama dili ile programlamayı öğreneceksiniz. Bu çerçeve altında kontrol sistemlerinde bilgi paylaşımı işlemlerinde önemli bir yere sahip olan LCD göstergenin programlanmasını, enerjiyi harekete çeviren servo motorun ileri seviye kontrollerini de öğreneceksiniz. Tüm bu yapıların elektronik ve otomasyon dünyasında çok önemli yerleri bulunmaktadır. Endüstriyel Otomasyonun yer aldığı hemen hemen her yerde karşılaşabileceğiniz bu sistemleri öğrenerek ve kontrol uygulaması gerçekleştirerek çok değerli tecrübeler elde edecek ve üst düzey tasarımları gerçeleştirebileceksiniz.

Mikrodenetleyici İle Lcd ve Servo Motor Kontrolu Modülü Mikrodenetleyici Uygulamaları Dersi'nin ikinci modülüdür. Mikrodenetleyicinin temel özelliklerini, temel PIC komutlarını ve temel uygulamaları daha önceki Mikrodenetleyici Programlama Temelleri modülünde öğrenmiştiniz. Bu modülde artık bilgilerinizi biraz daha ileri seviyeye taşıyacaksınız. Hazırlanan tüm öğrenme faaliyetleri birebir uygulamalı bilgileri içermektedir. Bu modül ile farklı yapıda tasarımlar gerçekleştirebilecek ve elektronik bilginizi geliştirebileceksiniz. Bu modülde verilen konuları öğrenirken internetten ve ilgili diğer elektronik kitaplardan da faydalanmanız size değişik bilgiler ve bakış açıları kazandıracaktır.

ÖĞRENME FAALİYETİ–1

AMAÇ

Mikrodenetleyici ile LCD kontrolünü hatasız olarak yapabileceksiniz.

ARAŞTIRMA

- PIC16F877 Mikrodenetleyici hakkında ön araştırma yapınız.
- LCD çeşitleri ve yapıları hakkında ön araştırma yapınız.

1. PIC16F877 İLE LCD KONTROLÜ

1.1. PIC16F877 Mikrodenetleyici

Bu modülde ise PIC ailesinin gelişmiş üyelerinden biri olan PIC16F877 ile uygulamalar yaptırılacaktır. Bu nedenle öncelikle PIC16F877'yi tanımak gereklidir.

Bu kısımda PIC16F84 ile ortak olan özellikler üzerinde ayrıntılı durulmayacak, aradaki farkları ortaya koyan bir yaklaşım tercih edilecektir.

PIC16F877'de PIC16F84'de olduğu gibi Harvard mimarisi kullanılmıştır. Veri yolu 8 bit genişliğindedir. Aynı anda veri belleğine 8 bit genişliğindeki bu yolla erişilirken; program belleğine program yolu ya da adres yolu denilen 14 bit genişliğindeki diğer bir yolla erişilir.

Bunun için PIC 16F877'de de komut kodları 14 bittir. 14 bitlik program belleğinin her bir adresi, bir komut koduna karşılık gelir.

1.1.1. PIC16F877 Mikrodenetleyicinin Özellikleri

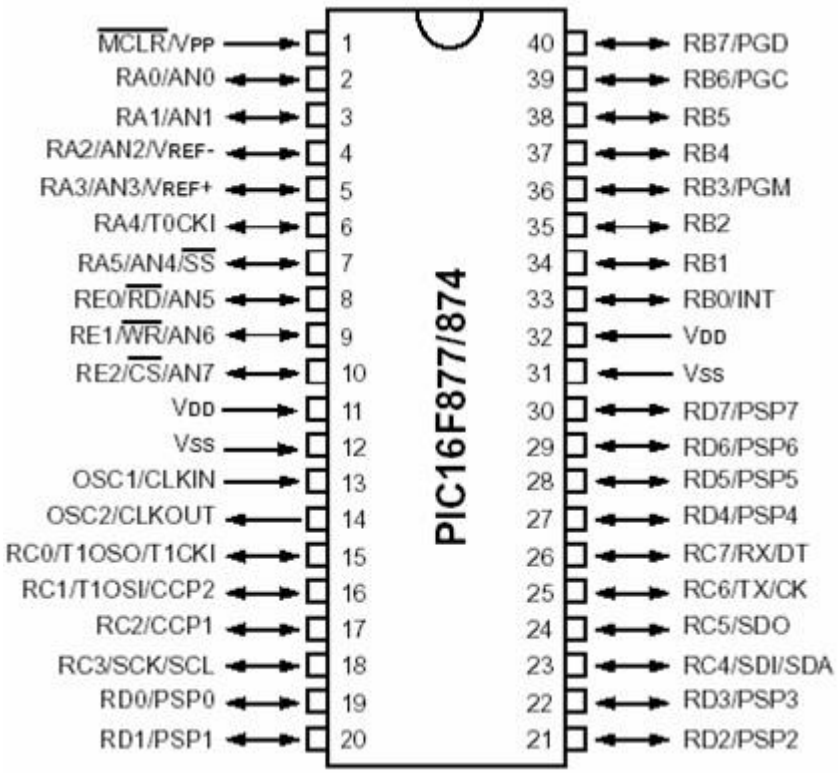
- CPU'su azaltılmış komut seti (RISC) temeline dayanır. Programlamada kullanılan
- 35 komut vardır ve her biri 14 bit uzunluktadır.
- Dallanma komutları iki saykıllık sürede, diğerleri ise bir saykıllık sürede uygulanır.
- İşlem hızı 20 MHz'e kadar arttırılabilir.
- Veri yolu 8 bittir.
- 8 KB flash program belleği vardır ve yaklaşık 1 milyon kez programlanabilir.
- 368 Byte veri belleği (RAM) bulunmaktadır.
- 256 Byte EEPROM veri belleği vardır.
- Port A,B,C,D,E olmak üzere 5 adet giriş çıkış portu bulunur.

- 54 adet SFR olarak adlandırılan özel işlem yazmacı vardır ve bunlar statik RAM üzerindedir.
- Pin çıkışları PIC 16C73B/74B/76 ve 77 ile uyumludur.
- 14 kaynaktan kesme yapabilir.
- Power-on Reset (Enerji verildiğinde sistemi resetleme özelliği) bulunmaktadır.
- Power-up Timer (Power-up zamanlayıcı) bulunmaktadır.
- Osilatör Start-up Timer (Osilatör başlatma zamanlayıcısı) bulunur.
- Watch-dog Timer (Bekçi köpeği zamanlayıcısı) bulunur.
- Programla kod güvenliğinin sağlanabilmesi özelliği vardır.
- Devre içi Debugger (Hata ayıklamakta kullanılabilecek modül) bulunur.
- Düşük gerilimli ve sadece 2 pinle programlama özelliği vardır.
- Enerji tasarrufu sağlayan uyku modu bulunmaktadır.
- Seçimli osilatör özelliklerine sahiptir.
- 2.0 V – 5.0 V arasında değişen besleme gerilimine sahiptir.
- 25 mA'lik kaynak akımı standardına sahiptir.
- Geniş sıcaklık aralığında çalışabilme özelliği vardır.
- Düşük güçle çalışabilme özelliği vardır.
- TMR0 8 bitlik zamanlayıcısı 8 bit önbölücülü olarak kullanılabilir.
- TMR1 önbölücülü 16 bit zamanlayıcı içeririr. Uyuma modundayken kontrol edilebilir ve değeri arttırılabilir.
- TMR2 8 bitlik zamanlayıcı hem önbölücü hem de son bölücü sabitine sahiptir.
- İki adet Yakalama / Karşılaştırma / PWM modülüne sahiptir. Yakalama ve karşılaştırma 16 bit, PWM ise maksimum 10 bit çözünürlükle yapılabilir.
- 10 bit çok kanallı A/D çeviriciye sahiptir.
- Ana Senkron seri port (MSSP) modülü, SPI (Master mod) ve I2C (Master Slave) modlarında kullanılabilir.
- Asenkron seri iletişim için USART seri iletişim ara birimine sahiptir.
- Paralel haberleşme için 8 bit genişlikte Paralel Slave Portu bulunur, bu port ile dış
- RD, WR, CS kontrollerine sahiptir.
- BOR Reset (Brown Out Reset) özelliğine sahiptir.
- PIC 16F84 ile PIC 16F877 mikrodenetleyiciler arasındaki farklar Şekil 1.1'de verilmiştir.

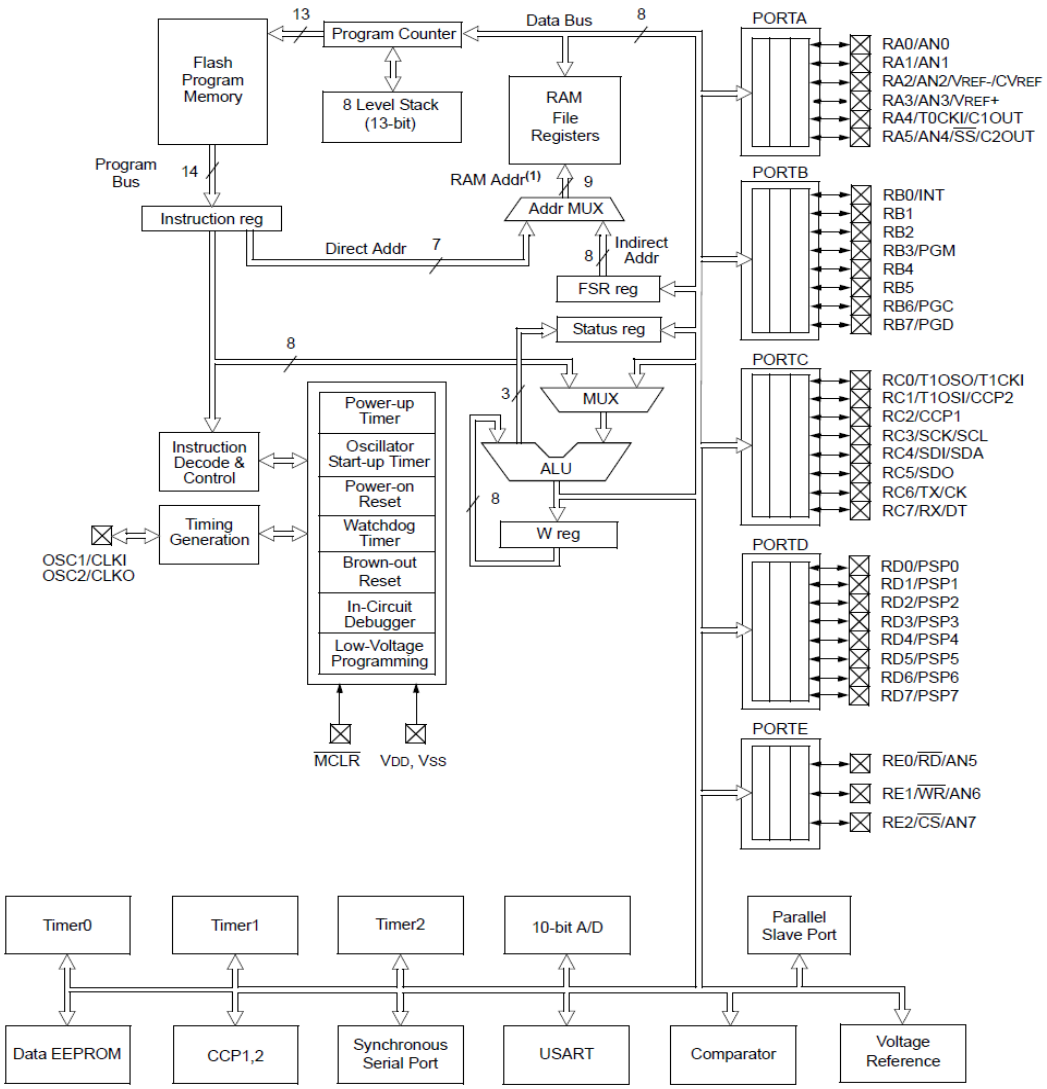
ÖZELLİKLER	PIC16F877	PIC16F84
Çalışma Hızı üst sınırı	20MHz	20MHz
Program Belleği	8 KB Flash ROM	1 KB Flash ROM
EEPROM Belleği	256 Byte	64 Byte
Kullanıcı RAM	368 Byte	68 Byte
Giriş/Çıkış Port Sayısı	33	13
Zamanlayıcı	Timer0, Timer1, Timer2	Timer0
A/D Çevirici	8 Kanal, 10 Bit	Yok
Capture/Compare/PWM	16 Bit Capture, 16 Bit Compare, 10 Bit PWM çözünürlük	Yok
Seri çevresel Arayüz	SPI (Master) ve 12C (Master - Slave) modunda SPI portu ve senkron seri port	Yok
Paralel Uydu Port (PSP)	8 Bit, harici, RD, WR ve CS kontrollü	Yok
USART/SCI	9 Bit adresli	Yok

Şekil 1.1: PIC16F84 ile PIC16F877 Arasındaki Farklar

1.1.2. PIC16F877 Mikrodenetleyicinin Fiziksel İç ve Dış Yapısı



Şekil 1.2: PIC16F877 Mikrodenetleyici Pin Tanımlamaları



Şekil 1.3: PIC16F877 Mikrodenetleyici iç yapısı

PİN ADI	PİN NO	PİN TİPİ	TAMPON TİPİ	AÇIKLAMALAR
OSC1/CLKIN	13	I	ST/CMOS(4)	Osilatör clock girişi (kristal veya harici kaynak)
OSC2/CLKOUT	14	O	-	Osilatör kristal çıkış ucu
$\overline{\text{MCLR}}$/Vpp	1	I/P	ST	Resetleme girişi / Programlama anında programlama gerilimi girişi (Mikrodenetleyicinin resetlenmesi için bu pin lojik 0 yapılmalıdır.)
RA0/AN0	2	I/O	TTL	Çift yönlü giriş/çıkış.Analog giriş 0.
RA1/AN1	3	I/O	TTL	Çift yönlü giriş/çıkış. Analog giriş 1.
RA2/AN2/V_{REF-}	4	I/O	TTL	Çift yönlü giriş/çıkış.Analog giriş 2. Negatif analog referans gerilimi.
RA3/AN3/V_{REF+}	5	I/O	TTL	Çift yönlü giriş/çıkış.Analog giriş 3. Pozitif analog referans gerilimi.
RA4/ T0CKI	6	I/O	ST	Çift yönlü giriş/çıkış. Bu pin istenirse TMR0 için bir clock girişi olabilir. Çıkış modunda açık drain tipindedir.
RA5/SS/ AN4	7	I/O	TTL	Çift yönlü giriş/çıkış. SSP için slave seçme pini. Analog giriş 4.
RB0/INT	33	I/O	TTL/ST(1)	Çift yönlü giriş/çıkış. Dış kesme girişi olarak seçilebilir.
RB1	34	I/O	TTL	Çift yönlü giriş/çıkış.
RB2	35	I/O	TTL	Çift yönlü giriş/çıkış.
RB3/PGM	36	I/O	TTL	Çift yönlü giriş/çıkış. Düşük gerilimli programlamada da kullanılabilir.
RB4	37	I/O	TTL	Çift yönlü giriş/çıkış. Portb kesme pini.
RB5	38	I/O	TTL	Çift yönlü giriş/çıkış. Portb kesme pini.
RB6/PGC	39	I/O	TTL/ST(2)	Çift yönlü giriş/çıkış. Portb kesme pini. Devre içi hata ayıklama pini. Seri programlamada saat darbesi girişi.
RB7/PGD	40	I/O	TTL/ST(2)	Çift yönlü giriş/çıkış. Portb kesme pini. Devre içi hata ayıklama pini. Seri programlamada veri girişi.

Tablo 1.4a: PIC16F877 Mikrodenetleyici pinlerinin fonksiyonları

PİN ADI	PİN NO	PİN TİPİ	TAMPON TİPİ	AÇIKLAMALAR
RC0/T1OS0/T1CK 1	15	I/O	ST	Çift yönlü giriş/çıkış. Timer1 osc. çıkışı veya Timer1 saat derbe girişi.
RC1/T1OS1/CCP2	16	I/O	ST	Çift yönlü giriş/çıkış. Timer1 osc. girişi. Capture2 girişi. Compare2 çıkışı. PWM2 çıkışı.
RC2/CCP1	17	I/O	ST	Çift yönlü giriş/çıkış. Capture1 girişi. Compare1 çıkışı. PWM1 çıkışı.
RC3/SCK/SCL	18	I/O	ST	Çift yönlü giriş/çıkış. SPI ve I^2C modları için senkron seri saat darbesi giriş/çıkışı.
RC4/SD1/SDA	23	I/O	ST	Çift yönlü giriş/çıkış. SPI modda SPI veri girişi veya I^2C modda veri girişi ya da çıkışı.
RC5/SDO	24	I/O	ST	Çift yönlü giriş/çıkış. SPI modda SPI veri çıkışı.
RC6/TX/CK	25	I/O	ST	Çift yönlü giriş/çıkış. USART asenkron gönderme veya senkron saat darbesi.
RC7/RX/DT	26	I/O	ST	Çift yönlü giriş/çıkış. USART asenkron alma veya senkron veri.
RD0/PSP0	19	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. Paralel Slave Port 0 (PSP0)
RD1/PSP1	20	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP1.
RD2/PSP2	21	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP2.
RD3/PSP3	22	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP3.
RD4/PSP4	27	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP4.
RD5/PSP5	28	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP5.
RD6/PSP6	29	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP6.
RD7/PSP7	30	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. PSP7.
RE0/RD/AN5	8	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. Analog giriş 5. Paralel slave port için okuma kontrol ucu.
RE1/WR/AN6	9	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. Analog giriş 6. Paralel slave port için yazma kontrol ucu.
RE2/CS/AN7	10	I/O	ST/TTL(3)	Çift yönlü giriş/çıkış. Analog giriş 7. Paralel slave port için seçme kontrol ucu.

Tablo 1.4b: PIC16F877 Mikrodenetleyici pinlerinin fonksiyonları

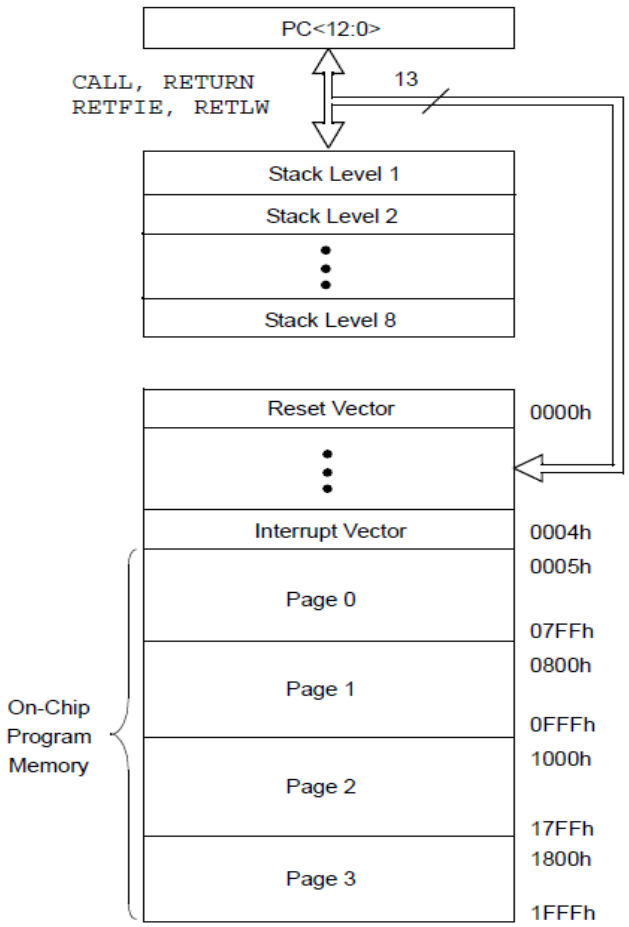
PİN ADI	PİN NO	PİN TİPİ	TAMPON TİPİ	AÇIKLAMALAR
V_{SS}	12, 31	P		Ground (toprak) ucu
V_{DD}	11, 32	P		Pozitif kaynak ucu
I: Input (Giriş) O: Output (Çıkış) I/O: Input/Output (Giriş/Çıkış) P: Power (Güç) TTL: TTL Giriş ST: Schimitt Trigger -: Kullanılmıyor Not: 1. Bu tampon portb dış kesme olarak düzenlendiğinde Schimitt Trigger girişidir. 2. Bu tampon seri programlama modunda Schimitt Trigger girişidir. 3. Bu tampon genel amaçlı giriş-çıkış olarak kullanıldığında Schimitt Trigger girişi ve PSP modunda TTL girişidir. 4. Bu tampon RC osilatör modunda Schimitt Trigger girişidir. Diğer modlarda CMOS girişidir.				

Tablo 1.4c: PIC16F877Mikrodenetleyici pinlerinin fonksiyonları

PIC16F877 Mikrodenetleyici içerisinde Harvard mimarisine göre tasarlanmış üç adet hafıza bloğu vardır.

1.1.3.1. Program Hafıza Organizasyonu

PIC16F877 içerisinde Şekil 1.3'te görüldüğü gibi 14 bitlik program yoluna bağlı 8KB'lık flash eprom program belleği (8Kx14) bulunmaktadır ve 13 bitlik program sayacı ile adreslenmektedir. Reset vektörü 0000h ve kesme vektörü de 0004h'dir. Şekil 1.5'te program hafıza haritası görülmektedir. Program kodları 0000h ile 1FFFh arasına yazılır. PIC16F877 ilk çalışmaya başladığı anda ya da herhangibir zamanda restlendiği anda program hafızada 0000h adresine gidilir. Kesme sinyali algılandığında ise program sayacı 0004h adresini üretir ve 0004h adresi üzerinden bir kesme alt programı çalışır.



Şekil 1.5: Program hafıza haritası

1.1.3.2. Veri Hafıza Organizasyonu

Veri hafızayı Flash Eeprom bellek ve RAM bellek olmak üzere iki kısımda inceleyebiliriz.

Flash Eeprom Bellek

256 Byte kapasiteli flash eeprom veri belleği 8 bitlik veri yoluna bağlıdır. Bu bellek kalıcı olarak veri kaydetmek amaçlı olarak kullanılır. Elektrik enerjisi kesildiğinde bile bu bilgiler bellekte saklanabilir ve daha sonra değiştirilebilir. Normal çalışma sırasında veri eeprom belleğe veri yazılabilir ya da eepromdan veri okunabilir. Bu bellek, diğer veri belleği gibi doğrudan adreslenemez. EEPROM veri belleğine dolaylı erişilir.

Eeprom bellek kullanımı ile ilgili altı adet özel fonksiyon yazmacı (SFR) bulunmaktadır. Bunlar EEDATA, EEDATH, EEADR, EEADRH, EECON1 ve EECON2 yazmaçlarıdır. Bunlarla ilgili bilgi ilerleyen kısımlarda verilecektir.

Ram Bellek

368 Byte ram bellek üzerinde genel ve özel amaçlı yazmaçların yer aldığı dört adet saklayıcı kümesi(bank) yer almaktadır. Status yazmacının 5. ve 6. bitleri ile küme seçimi yapılmaktadır. Her bank 128 baytdır ve düşük kelime adreslerinde özel fonksiyon yazmaçları yer almakta iken yüksek adresli değerlerde genel amaçlı yazçmaçlar yer almaktadır. Özel fonksiyon yazmaçlarının bazıları daha hızlı erişim için birden fazla kümede yer almaktadır.

Tablo 1.6'da PIC16F877 Mikrodenetleyicisinin RAM haritası görülmektedir.

	File Address		File Address		File Address		File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876A.
2: These registers are reserved; maintain these registers clear.

Tablo 1.6: PIC16F877 RAM Haritası

Açıklamalar:

- Fiziksel bir yazmaç değildir.
- PIC16F876'da kullanılmayan bir yazmaçtır
- Bu yazmaçlar kullanılamaz. Lojik-0 değerinde kalmalıdır.

Tablo 1.6'da görülen GPR bölgesine doğrudan ya da dosya seçim yazmacı (FSR) üzerinden dolaylı olarak ulaşılabilir. Özel fonksiyon yazmaçları mikroişlemci ve çevresel modüller tarafından özel işlemleri kontrol etmek için kullanılır. Bu modülde gerektiğinde uygulamalarda kullanılacak özel fonksiyon saklayıcıları ilgili bilgiler uygulama çalışmalarında verilecektir.

Yazmaçların İşlevleri:

1. PORTA PORTE

Portlar mikrodenetleyicinin dış dünyadan bilgi alması ve kendi dışındaki devrelere veri aktarabilmesi amacıyla kullanılır. PIC16F877'nin beş portu vardır. A portu 6 bit genişliğindedir. B, C, D portları 8 bit, E portu ise 3 bit genişliğindedir.

2. TRISA TRISE

Portların yönünü belirleyen yazmaçlardır. Eğer portların herhangi bir pininden mikrodenetleyici dışına veri gönderilecekse, önce ilgili portun yön yazmacı aynı numaralı biti lojik-0 yapılır. Eğer herhangibir pinden mikrodenetleyiciye veri girilecekse, yine önceden, o portun yön yazmacının aynı numaralı biti lojik-1 yapılır.

3. TMR0, TMR1 ve TMR2

Mikrodenetleyici içinde bulunan zamanlayıcı ve sayaç olarak çalıştırılan bölümü denetleyen yazmaçlardır. TMR1 16 bitlik bir zamanlayıcıdır. TMR1L ve TMR1H olmak üzere iki adet 8 bitlik saklayıcıdan oluşur. TMR0 ve TMR2 8 bitliktir. TMR2 PWM sinyalinini zamanlamasında da kullanılır.

4. T1CON, T2CON ve PR2

Timer1 ve Timer2 zamanlayıcılarını kontrol etmek için kullanılırlar. PR2 Timer2 zamanlayıcısının son sayma değerini diğer bir deyişle 00h'e döndüğü değeri saklar.

5. EEDATA ve EEDATH, EEADR ve EEADRH, EECON1 ve EECON2

Mikrodenetleyici içindeki EEPROM veri belleğine ulaşmakta kullanılır. EEADR yazmacında adres numarası bulunan veri, EEPROM veri belleğinden okunarak EEDATA yazmacına getirilir. EEADR yazmaçı eprom bellekte erişilmek istenen adresi tutar.

EEDATA ise yazılmak ya da okunmak istenen veriyi tutar. EEADRH 13 bitlik adresin ve EEDATH 14 bitlik verinin yüksek değerli bitlerini tutmak için kullanılır. EECON1 yazmacı erişimi başlatmak ve ayarlamak için kullanılan bir kontrol yazmacıdır. EECON2 katalog bilgisinde fiziksel olarak kullanılmayan bir yazmaç olarak geçmektedir. Donanımsal olarak yazma işlemi sırasında hatalı yazım işlemlerinden korunma amaçlı olarak kullanılmaktadır.

6. GPR

Genel amaçlı yazmaçların adresleri yukarıdaki çizelgede verilmişti. Programcı buradaki adresleri istediği gibi, kendi değişkenleri için kullanabilir. Bu adreslere isterse programın içinde özel adlar verebilir.

7. Durum Yazmacı

Durum(status) yazmacı, aritmetik ve mantık biriminin(ALU), aritmetik işlem sonucundaki durumunu, merkezi işlem biriminin (CPU) test durumlarını ve veri belleğine ait küme(bank) seçme bitlerini tutar. Herhangi bir yazmaç gibi içeriği okunabilir ve değiştirilebilir. Ancak 3. ve 4. bitleri sadece okunabilir, değiştirilemez.

Eğer, bu yazmacın içeriği CLRF STATUS komutuyla, silinmek istenirse sadece üst üç bit lojik-0 olur. Bu komut sonunda STATUS'un içeriği 000uu1uu değerini alır. Burada u:değişmez (Unchangable) anlamındadır.

7	6	5	4	3	2	1	0
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
R/W	R/W	R	R	R/W	R/W	R/W	R/W

Şekil 1.7: Durum yazmacı

Bit 7: IRP: Yazmaç bank seçme biti (dolaylı adreslemede kullanılır)
0= Bank 0,1 (00h-FFh)
1= Bank 2,3 (100h-1FFh)
Bit 6-5: RP1:RP0: Yazmaç bank seçme biti (doğrudan adreslemede kullanılır)
00= Bank 0
01= Bank1
10= Bank 2
11= Bank 3
Her bir bank 128 byte'tır.
Bit 4: $\overline{TO}$: Time-out (süre aşımı biti)
1= CLRWDT komutuyla veya SLEEP'den güç verme durumuna geçirildiğinde 1 olur.
0= WDT time-out (süre aşımı) işlemi gerçekleşmiş ise 0 olur.

Bit 3: $\overline{PD}$: Power-down (Güç kesme) biti
1= CLRWDT komutu ile veya güç verme durumunda 1 olur.
0= SLEEP komutu çalıştırılınca 0 olur.

Bit 2: Z: Zero (Sonuç sıfır) biti
1= Aritmetik veya lojik işlem sonucu 0 ise bu bit 1 olur.
0= Aritmetik veya lojik işlem sonucu 0 değil ise bu bit 0 olur.

Bit 1: DC: Dijit elde (Digit Carry/Borrow) biti. (ADDWF, ADDLW komutları için.)
1= 8 Bitin düşük öncelikli dörtlüsü, taşarsa bu bit lojik 1 olur.
0= 8 Bitin düşük öncelikli dörtlüsü taşmazsa, bu bit 0 olur.

Bit 0: C: Carry/Borrow biti (ADDWF ve ADDLW komutları için)
1= 7.bitten sonra taşma olursa bu bit 1 olur.
0= 7.bitten sonra taşma olmazsa bu bit 0 olur.

8. Option Register

Option Register, okunabilir ve yazılabilir bir yazmaçtır. Kapsamında TMR0 / WDT zamanlayıcılarının konfigürasyon bitleri, dış kesme (interrupt) denetim bitleri, TMR0 zamanlayıcısı kesme denetim bitleri ve PORTB için yükseğe çekme (pull-up) dirençlerinin kullanılmasını sağlayan bit bulunur.

9. INTCON

INTCON okunabilir ve yazılabilir bir yazmaçtır. Kapsamında TMR0 / WDT yazmacı taşma uyarı bitleri, RB port değişim ve dış kesme (RB0/INT pin interrupt) denetim bitleri, TMR0 kesme denetim bitleri bulunur.

10. PIE1

PIE1 çevresel kesmelerle ilgili bitleri olan bir yazmaçtır. Bir çevresel kesmenin olabilmesi için, PIE1 (INTCON<6>) biti de set edilmelidir.

11. PIR1

PIR1 çevresel kesmelerle ilgili uyarı bitlerini taşıyan yazmaçtır.

12. PIE2

PIE2 yazmacı CCP2 (Capture/Compare/PWM2) çevresel biriminin kesme bitlerini, SSP (Senkron Seri Port) veri yolu çarpışma bitini ve EEPROM yazma kesmesi bitini taşır.

13. PIR2

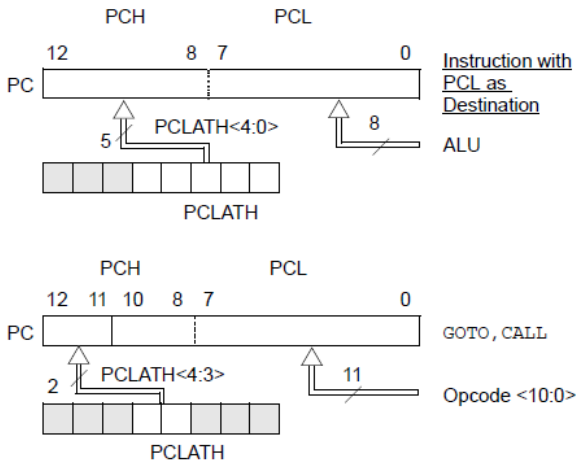
PIR2 çevresel kesmelerle ilgili diğer uyarı bitlerini taşıyan yazmaçtır.

14. PCON

Güç kontrol yazmacı olan PCON, yazılımda ve reset durumlarında kullanılır. Reset durumları; devrenin dışardan MCLR ile, gerilim ya da akımın aşırı düşme ve yükselmesi Brown-Out, Watch Dog Timer ve son olarak Power on reset durumlarında kullanılabilir. BOR biti, Power on reset'te bilinemez. Bir sonraki BOR durumunun öğrenilebilmesi için reset sonrasında lojik-1 yapılmalıdır.

15. PCL ve PCLATH

Program sayacı (PC) olarak adlandırılan adresleme yazmacı 13 bitliktir. Bunun düşük değerlikli byte'ı PCL yazmacından gelir. Üstteki bitler ise PC<12:8> arasındaki 5 bittir, bunlar PCLATH yazmacından alınır. PCL okunabilir ve yazılabilir bir yazmaçtır. Ancak üst bitleri (PCH) doğrudan okunamaz. Dolaylı olarak PCLath yoluyla yazılabilir veya okunabilir.



Şekil 1.8: PC ve PCLATH Yazmaçları

Call komutu, yığının her zaman en tepesine, PCL yazmacının içindeki adres değerini yazar. Return, Retfie ve Retlw komutları ise yığının en tepesindeki elemanın içeriğini PCL'ye aktarır. Saklayıcı küme numaralarının PCLath yazmacından PC'ye aktarılabildiği program yazarken de unutulmamalıdır. PCLATH yazmacının içeriği, alt programa girildikten sonra sabit kalır, bir return ya da retfie benzeri komut gelse de değişmez. Programcı, call veya goto komutlarından önce, PCLATH yazmacını güncellemelidir. PCH daima PCLATH yazmacı yoluyla güncellendiğinden (tersi yapılamaz) alt program veya gidilen kesimin hangi kümede olduğu, aşağıdaki örneğe benzer bir yolla belirtilmelidir.

```
ORG 0x500
BCF PCLATH, 4 ;PCLATH yazmacının 4.biti temizlendi
BSF PCLATH, 3 ;PCLATH'ın 3.biti set edildi, 1.bank geçildi
;(800h-FFFh adres aralığı)
CALL SUB1_P1 ;1.banktaki alt program çağrıldı
ORG 0x900 ;Bank 1 (800h-FFFh)
SUB1_P1
Altprogram 800h ile FFFh aralığına yerleştirildi.
RETURN ; Return'den sonra 0.sayfaya (000h-7FFh) dönülecek
```

16. YIĞIN Bölgesi

Yığın bölgesi 8 yazmaçtan oluşur. Her yazmaç 13 bitliktir ve donanımın bir parçasıdır. Veri veya program alanlarında yer almaz. Yığın göstergesi yazılabilir ve okunabilir değildir. Yığın işlemi komutları POP ve PUSH'tur. Her PUSH işleminde yığının en tepesindeki adrese, PC'ın içeriği yüklenir. Her POP işleminde yığının en tepesindeki adres PC'ın içine geri yüklenir. Yığın, LIFO (Last In First Out) son giren ilk çıkar tekniğiyle çalışır.

17. INDF ve FSR

INDF fiziksel bir yazmaç değildir. Mikrokontrolördeki RAM adresini tutar. INDF'e yazılan her veri, adresi FSR yazmacında bulunan RAM'a yazılır. INDF'ten okunan veriler de adresi FSR'de bulunan RAM'den okunmuştur.

18. SSPBUF, SSPCON, SSPCON2, SSPSTAT ve SSPADD

Senkron seri port alma ve verme sırasında veri SSPBUF yazmacına yazılır. SSPCON ve SSPCON2 ise senkron seri port haberleşmesini kontrol eden yazmaçlardır. Senkron seri portun durum bitleri SSPSTAT yazmacında kayıtlıdır. SSPADD senkron seri port adres yazmacıdır.

Dolaylı adreslemede INDF ile birlikte kullanılır. Mikrodenetleyicinin içindeki RAM adresinde yapılacak işlemlerde, RAM adresini tutar. Bu durumda INDF'ye yazılacak her veri, aslında adresi FSR'de bulunan RAM'e yazılmıştır.

19. CCPR1L, CCPR1H, CCP1CON, CCPR2L, CCPR2H ve CCP2CON

Yakalama-Karşılaştırma-PWM işlemleri ile ilgili yazmaçlardır. Bu işlemler için PIC16F877'de iki ayrı grup bulunmaktadır. CCP1CON ve CCP2CON yazmaçları control bitlerinin bulunduğu yazmaçlardır. CCPR1L ve CCPR2L ilk 8 biti saklarlar. CCPR1H ve CCPR2H yüksek değerli 8 biti saklar.

20. TXSTA, RCSTA, SPBRG, TXREG ve RCREG

Bu yazmaçlar üniversal asenron seri haberleşmede kullanılır. TXSTA seri bilgi vermede durum ve kontrol bitlerinin yer aldığı yazmaçtır. RCSTA Seri bilgi almada durum ve kontrol bitlerinin yer aldığı yazmaçtır. SPRG ise baud hızını belirleyen yazmaçtır. TXREG USART bilgi vermede veri yazmacı olarak kullanılır. RCREG ise USART bilgi almada veri yazmacı olarak kullanılır.

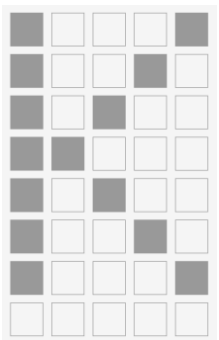
21. ADRESH, ADRESL, ADCON0, ADCON1

ADRESH A/D çevrim sonucunun yüksek değerli 8 bitini, ADRESL ise düşük değerli 8 bitini saklar. ADCON0 ve ADCON1 ise A/D çevrimi kontrol eden saklayıcılardır.

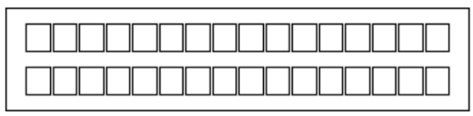
1.2. Lcd Yapısı ve Programlanması

1.2.1. LCD'lerin Genel Özellikleri ve Çeşitleri

Liquid **C**rystal **D**isplay kelimelerinin ilk harfleri alınarak LCD kısaltması yapılmış olup sıvı kristal görüntü birimi olarak Türkçe'ye çevrilebilir. LCD ler ekran tipi bakımından karakter ve grafik olmak üzere ikiye ayrılır. Karakter tabanlı LCD ler sembol, harf ve karakterleri görüntüler. Karakter LCD' ler 5 x 8 veya 5 x 11' luk dot matris (noktasal matris) hücrelerinden oluşur. Her bir hücre ışık veren LED diyot gibi düşünülebilir. En alt satır imleç için kullanıldığından bu LCD'ler 5 x 7 veya 5 x 10 olarak söylenir. Şekil 1.9'da 5 x 7'lik dot matris LCD'de K karakteri görülmektedir.



Şekil 1.9: Dot matris LCD



Şekil 1.10: 2x 16 LCD

LCD' karakterlerin gösterilmesi için 7 segment displaylerde olduğu gibi tarama yöntemi kullanılır. LCD üzerinde tarama işlemini yapan entegreler bulunur. Karakter LCD'ler satır ve sütun sayısı ile tanımlanır. Sütun sayıları 8-16-20-24-32-40 olarak yapılır. Satır sayıları ise ise 1,2,4 ya da daha çok satırlı olabilir. 2x16 denince 2 satırlı 16 sütunlu LCD akla gelmelidir. Şekil 1.10 'da görülen 2 x 16 LCD üzerinde toplam olarak 32 karakter gösterilebilir.

Grafik LCD'ler karakter tabanlı LCD lerin tüm özelliklerini yerine getirebilir. Şekil 1.11'de bir grafik LCD görülmektedir. Piksel düzeyinde kontrol imkanı olduğundan istenen çizimleri de gösterebilir. Renkli ve siyah beyaz, dokunmatik tipleri bulunmaktadır. Ekran çözünürlükleri ile belirtilir, örneğin 128x64 pixel çözünürlüğe sahip bir LCD 8192 pikselden oluşur. Grafik LCD lerin kontrol işlemi karakter LCD'lere göre daha zordur.

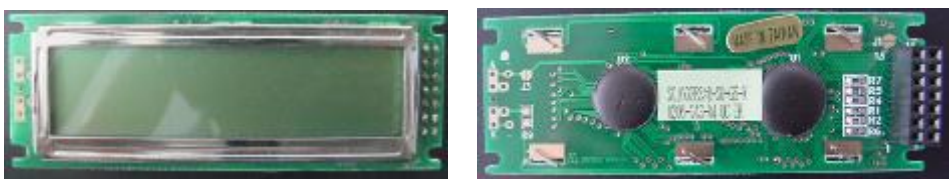
Şekil 1.11: Grafik LCD

LCD'ler bilginin uygulanış yöntemine göre paralel ve seri girişli olmak üzere iki kısımda incelenebilmektedir. Paralel LCD lerde 4 bit veri iletim modunda toplam 7 pin kullanılmak zorundadır. Seri LCD de veri iletimi için sadece bir pine ihtiyaç vardır. Yaygın kullanımı nedeniyle bu modülde paralel karakter LCD üzerinde durulacaktır.

1.2.2. Paralel Karakter LCD Yapısı

1.2.2.1.Paralel LCD Yapısı

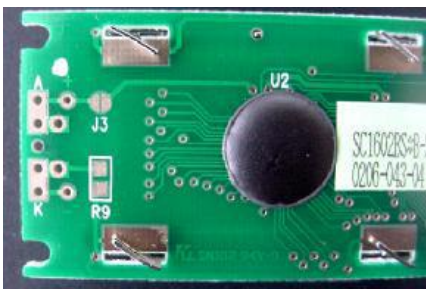
Şekil 1.12 'de HD44780 paralel girişli LCD görülmektedir. HD44780 LCD devresinde yer alan epoksi damla olarak yerleştirilmiş sürücü entegresidir. Bu kısımda piyasada bulunması kolay ve kullanım alanı geniş olduğundan dolayı bu LCD ile ilgili bilgiler verilip uygulama yapılacaktır. HD44780 işlemci LCD'ler için standart olmuş bir işlemcidir. LCD'nin arkasına bakılıp işlemci seri numarası okunabilir ve işlemcinin yapısı kataloglardan öğrenilebilir. Günümüzde birçok firmanın ürettiği LCD, HD44780 ile uyumludur. LCD bilgisayarla veya mikrodenetleyici ile kontrol edilebilir. 2x8 ve 1x16 LCD'de sadece bir işlemci kullanılırken, 2x16 ve 2x20 LCD'lerde işlemcinin yanı sıra bir de hafıza entegresi bulunur. Boyut arttıkça hafıza kapasitesi de artar. Tüm entegreler siyah epoksi damla şeklinde veya entegre şeklinde PCB üzerine yerleştirilir.



(a) Ön görünüş **(b) Arka görünüş**

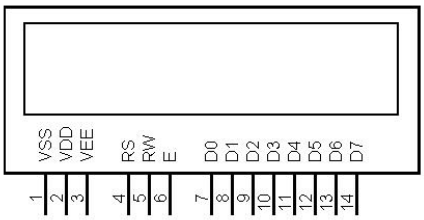
Şekil 1.12: HD44780 LCD Display

LCD'nin dış dünya ile haberleşmesi için pinleri bulunmaktadır. HD44780 uyumlu LCD'de 14 pin bulunmaktadır. Bu pinler 2'şerli 7 sıra halinde olabileceği gibi 14'lü tek sıra olabilir. Bu pinlerin fonksiyonları Şekil 1.15'teki tabloda görülmektedir. LCD 4x40 ise ilave bir pini daha vardır. Ayrıca bazı LCD'lerde ışıklandırma amacıyla 15. ve 16. pinlere de rastlanabilir. 15. ve 16. pinler yoksa LCD devresinde aydınlatma uçlarından da bağlantı yapılabilir. Bu ilave pinlerin karakter gösterilmesinde bir etkisi yoktur. Şekil 1.13'te ışıklandırma uçları A-K görülmektedir. A (Anot) olup pozitif voltaj ucudur. K (Katot) olup negatif voltaj ucudur.



Şekil 1.13: HD44780 LCD displayde ekran ışığı bağlantı uçları

Şekil 1.14 te LCD nin ayak bağlantıları görülmektedir. Şekil 1.14 te verilen LCD nin pin isim ve numaraları piyasada bulunan bir çok LCD ile uyumludur. LCD nin arkasında yazan modeli veya LCD nin kullandığı entegre kodu ile internetten ilgili dökümanlara ulaşılarak LCD uçlarının karşılıkları öğrenilebilir.



Şekil 1.14: 2x16 LCD Pin İsimleri

Şekil 1.15 te LCD pin numaraları ve görevi tablo halinde verilmiştir.

Pin No	Görevi
1	Vdd (5V gerilim ucu)
2	Vss (Şase bağlantı ucu)
3	Vee (Kontrast gerilimi ucu)
4	RS (Veri /Komut seçim ucu)
5	R/W (Okuma- Yazma ucu)
6	E (Uygulama izin ucu)
7	D0 (0. veri (LSB) ucu)
8	D1 (1. veri ucu)
9	D2 (2. veri ucu)
10	D3 (3. veri ucu)
11	D4 (4. veri ucu)
12	D5 (5. veri ucu)
13	D6 (6. veri ucu)
14	D7 (7. veri (MSB) ucu)

Şekil: 1.15 : LCD pin numaraları ve görevi .

LCD Pin'lerinin Görevinin Açıklanması;

Vdd ve Vss: LCD Vdd pozitif besleme ucudur. Bu ucu +5V gerilim uygulanır. Vss ucu şase ucudur.

Vee: Bu uç ile LCD ekranın kontrast seviyesi değiştirilebilir. Bu uç bazı kaynaklarda V_0 ismiyle gösterilir.

RS (**R**egister **S**elect) **-** (Veri /Komut ucu)**:** RS=1 ise D0-D7 uçlarından (Veri yolu) gelen bilgi karakter/veri olarak algılanır. RS=0 olursa veri hattındaki bilgi komut olarak algılanır.

R/W (Read/Write): LCD ye bilgi göndermek ve LCD den bilgi almak için kullanılır. Lojik 1 ise hafıza hücresinden veri okunur. Lojik 0 ise LCD hafıza hücresine veri yazılır.

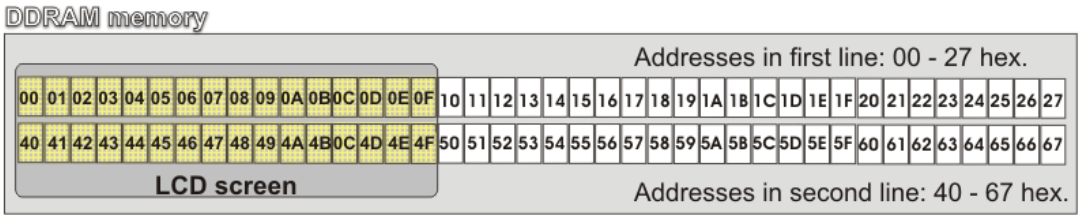
E (Enable): Yetki verme ucudur. Gönderilen komutun uygulanması için veya verinin işlenmesi için kontrol devresinden bu uca lojik-1'den 0'a düşen (düşen kenar) bir kare dalga sinyal gönderilir.

D0..D7 uçları: Komut girişi ve veri giriş-çıkışı için kullanılırlar. 8 bitlik modda tümü kullanılırken 4 bitlik LCD çalışmasında D7..D4 uçları kullanılır.

LED (A) ve LED (K) uçları: Bu uçlar arka ışıklandırmaya sahip LCD lerde bulunur. LED (A) ucuna +5V, LED (K) uçlarına şase uygulanır.

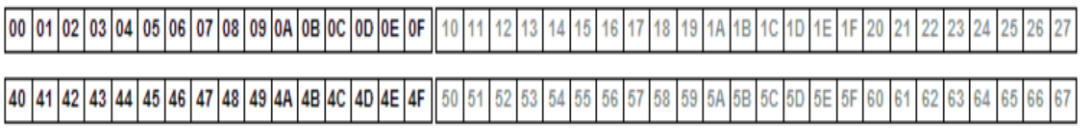
1.2.2.2.Paralel LCD Bellek Yapıları

LCD'lerde DDRAM ve CGRAM olmak üzere iki adet RAM hafıza hücresi yer alır. DDRAM (**D**isplay **D**ata **RAM**) displayde gösterilmesi istenen karakter kodlarının tutulduğu geçici veri belleğidir. LCD'nin satır ve sütunlarına karakter yazmak için kullanılır. Şekil 1.16 da görüldüğü gibi 80 Baytlık veya diğer tabirle 80 adet karakter kapasitesine sahiptir. DDRAM adres değerleri hex formatında gösterilmiştir.



Şekil 1.16: DDRAM Adresleri

DDRAM içeriğinin LCD'de görünen kısmı LCD'nin satır ve sütun sayısı ile sınırlıdır. Örneğin 2x16' lık bir LCD'de birinci satırda 16, ikinci satırda 16 olmak üzere toplam 32 karakter ekranda görülebilir. Şekil 1.17'de DDRAM adreslerinin LCD de görünen kısımları koyu renkli, görünmeyen kısımları ise açık renkli olarak gösterilmiştir. Şekil 1.17' de görüleceği üzere DDRAM 'in 00 h ile 0F h arasındaki adresleri LCD nin 1. satırında satırında, 40 h ile 4F h arasındaki adresleri LCD nin 2. satırında görüntülenir.

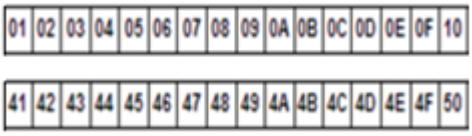


Şekil 1.17: DDRAM Adreslerinin LCD de görünen ve görünmeyen kısımları

Diğer adresleri görmek için DDRAM sağa veya sola kaydırılır. Bu işlemi somutlaştırmak için hayalimizde bazı işlemler yapalım, ilk olarak DDRAM adreslerin yazılı olduğu kağıt şeriti dünya küresel haritasının tamamını çevreleyecek şekilde yapıştıralım, yapıştırmadan önce şeritin sağ ve sol kenarlarının bitişik durmasına özen gösterilim, sonra adres şeriti üzerinde 2 satır ve 16 sütun görünecek şekilde bir çerçeve hazırlayalım,

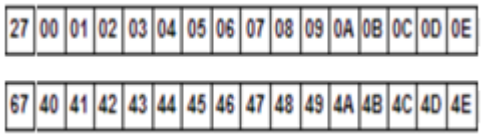
çerçeveyi sabit tutup dünya küresini sağa veya sola çevirelim. Çerçeve içerisinde görünen adresler LCD ekranında görüntülenecek olanlardır.

Şekil 1.18' de DDRAM içeriği 1 kez sola kaydırılmıştır. Şekil 1.4'ten görüleceği üzere DDRAM'in 10 h ve 50 h adresleri LCD de görünür duruma gelmiştir. 00h ve 40h adresleri görünür durumdan çıkmıştır.



Şekil 1.18: DDRAM içeriğini 1 kez sola kaydırıldığında 2x 16 LCD görünümü

Şekil 1.19' te DDRAM içeriği 1 kez sağa kaydırılmıştır. Şekil 1.19'dan görüleceği üzere 27 h ve 67 h adresleri LCD de görünür duruma gelmiştir. 0F ve 4F adresleri görünür durumdan çıkmıştır.



Şekil 1.19: DDRAM içeriğini 1 kez sağa kaydırıldığında 2x 16 LCD görünümü

LCD'nin içinde bulunan diğer hafıza ise CGROM (**C**haracter **G**enarator **R**OM)'dur. Karakter üretici ROM bellek olarak tanımlanır ve 192 Baytlık bir hafızadır. CGROM içerisinde ASCII karakter tablosunda yer alan standart karakterler kayıtlıdır. Toplam karakter sayısı 192'dir. Bunlardan 64 tanesi genel olarak kullanılan ascii karakterleri, 64 tanesi Japon karakterleri ve 32 tanesi de özel yunan alfabesi karakterleridir. Bu karakterlerden genel olarak kullanılan 64 tane standart karakter

Şekil 1.20'de ASCII karakter tablosunda görülmektedir. Bu tabloda verilen sabit karakterler değiştirilemez. Fakat kullanıcılar isteğe bağlı olarak 8 adet karakter daha tanımlayabilir. Bu tanımlanan karakterler LCD de CGRAM (**C**haracter **G**enarator **R**AM) de saklanır. Tabloda dikkat edilirse 0x00 ile 0xFF' e kadar olan ilk 16 alan boştur. Bu 16 alana kullanıcılar istediği karakteri kaydedebilir. İleriki uygulamalarda bu alanda Türkçe karakterler oluşturulacaktır.

Şekil 1.20: LCD ASII karakter tablosu

LCD de gösterilmek istenen karakter verileri **AC adres sayıcı ile (A**dres **C**ounter-**AC)** yerleştirilir. Şekil 1.21' de AC adres sayıcı görülmektedir.

AC6	AC5	AC4	AC3	AC2	AC1	AC0

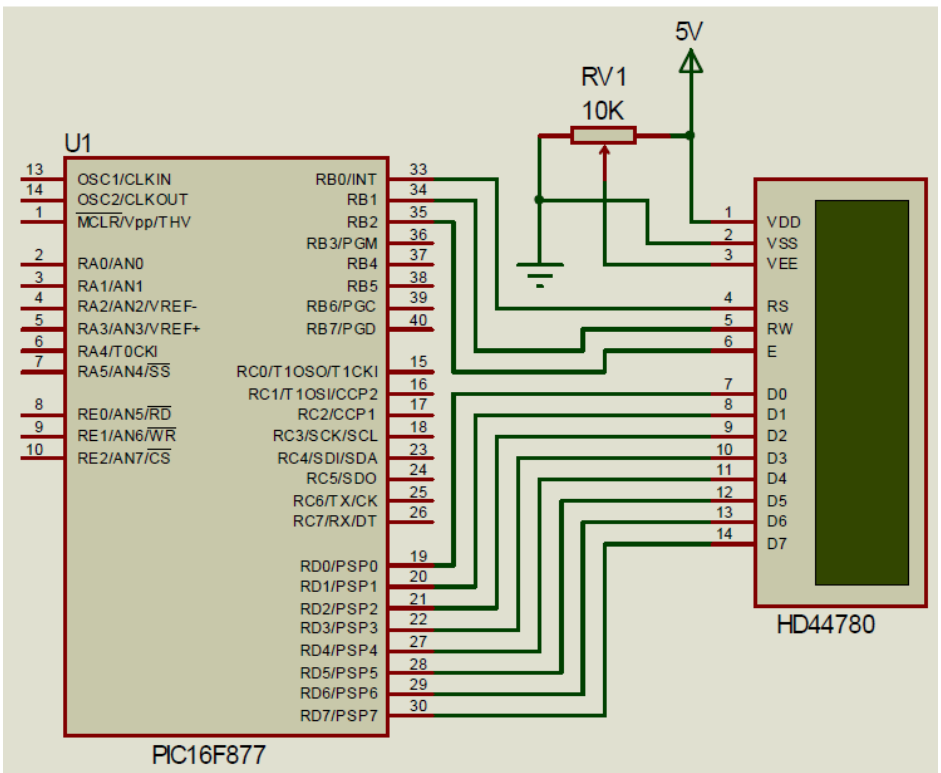
Şekil 1.21: AC adres kaydedicisi (Adres Counter-AC)

Örnek olarak DDRAM' in 01 h adresine veri yazmak veya okumak istenirse AC adres kaydedicisine 0000001 değeri yazılır. DDRAM in 5D h adresine veri yazmak veya okumak istenirse AC adres kaydedicisine 1011101 değeri yazılır. LCD ye veri gönderilirken AC sayıcı içeriği otomatik olarak artar. LCD ye gönderilen veriler 00h adresinden başlanarak 67 h adresine kadar otomatik olarak yerleştirilir.

AC adres kaydedicisi CGRAM (Character Genarator RAM) içerisine veri yazmak veya okumak içinde kullanılır.

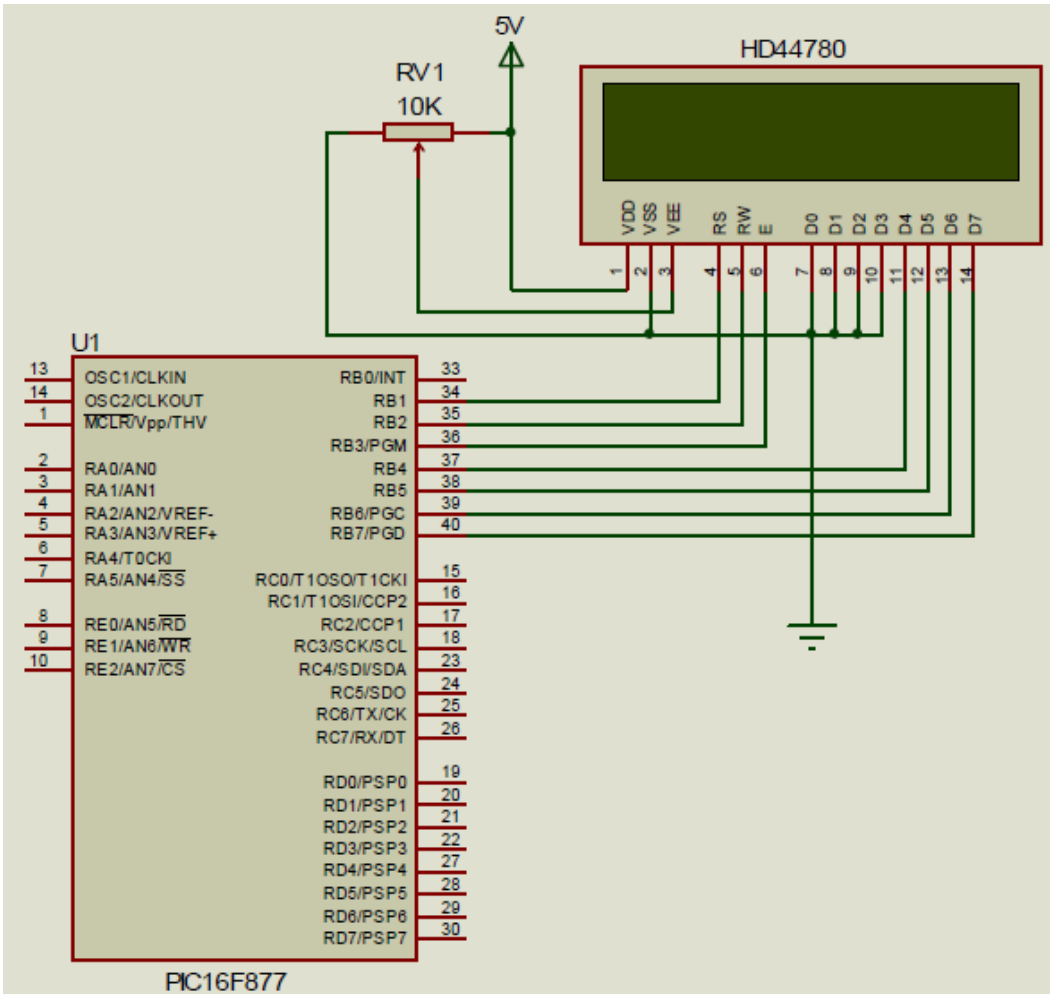
1.2.2.3. Paralel LCD Bağlantı Modları

Şekil 1.22'te PIC16F877 ile LCD bağlantısı için örnek bir devre görülmektedir. PortB'nin ilk üç biti kontrol bitleri için kullanılmış, Port D ise veri bitleri için kullanılmıştır. RV1 potansiyometresi ile LCD'nin kontrastı ayarlanmaktadır. Bu bağlantı ile LCD 8 bit modda kontrol edilebilir. Ancak yazılan programda program kodları 8 bitlik çalışmaya göre yazılmalıdır.



Şekil 1.22: PIC16877-LCD Bağlantısı (8 bit Modu)

Şekil 1.23' te PIC16F877 mikrodenetleyicinin 4 bitlik moda LCD'yi kontrol ettiği örnek bir devre görülmektedir. Bu devrede LCD'nin 4 bit modda bağlantısı için sadece PortB kullanılmıştır. Böylece mikrodenetleyicinin diğer portları farklı amaçlar için kullanılabilir.



Şekil 1.23: PIC16877-LCD Bağlantısı (4 bit Modu)

1.2.2.4. HD44780 Tabanlı LCD Komutları

KOMUT	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	KOMUT ICRA SÜRESI
Ekran sil- Clear display	0	0	0	0	0	0	0	0	0	1	1.64mS
İmleç Başa- Cursor home	0	0	0	0	0	0	0	0	1	x	1.64mS
Giriş Modu- Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40uS
Display aç/kapa ve imleç kontrol- Display on/off &cursor control	0	0	0	0	0	0	1	D	U	B	40uS
İmleç/ Display kaydır- Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	40uS
Fonksiyon Ayarları- Function set	0	0	0	0	1	DL	N	F	x	x	40uS
CGRAM adres seçme-Set CGRAM address	0	0	0	1	CGRAM address						40uS
DDRAM adres seçme-Set DDRAM address	0	0	1	DDRAM address							40uS
Meşgul Bayrağı ve Adres oku - Read “BUSY” flag (BF)& address	0	1	BF	CGRAM veya DDRAM address							-
CGRAM veya DDRAM a yazma- Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	40uS
CGRAM veya DDRAM dan okuma- Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	40uS

Tablo 1.24: LCD Komutları

Tablo ile ilgili açıklamalar:

- DDRAM: Display Data RAM
- CGRAM (**C**haracter **G**enarator **R**AM): Karakter Üretici RAM
- BF (**B**usy **F**lag) = 1 ise komut kabul edilmez. LCD meşguldur. BF=0 ise komut kabul edilir.
- N (**N**umber of display lines) = Display satır sayısını gösterir. Lojik 1 ise 2 satır kullanılır, Lojik 0 ise 1 satır.
- X = Önemsiz giriş 1 ya da 0 olabilir.
- F (**F**ont) = Lojik 1 ise 5x10 karakter fontu, Lojik 0 ise 5x7 karakter fontu kullanılır.
- I/D (**I**ncrement/ **D**ecrement) = Lojik 1 ise kursör pozisyonunu 1 arttırılır, Lojik 0 ise kursör pozisyonunu 1 azaltılır.
- S (**S**hift) = Lojik 1 ise display kayar, Lojik 0 ise display kaymaz.
- S/C (**S**hift/ **C**ursor) = Lojik 1 ise display kaydırılır, Lojik 0 ise imleç kaydırılır
- R/L (**R**ight/**L**eft)= Lojik 1 ise sağa kaydırılır, Lojik 0 ise sola kaydırılır
- D (**D**isplay) = Lojik 1 ise display açılır, Lojik 0 ise display kapatılır.
- C (**C**ursor) = Lojik 1 ise imleç gösterilir, Lojik 0 ise imleç gösterilmez
- B (**B**link)= Lojik 1ise imleç yanıp söner, Lojik 0 ise yanıp sönmez
- DL (**D**ata **L**ength)= Lojik 1 ise veri yolu 8 bit, Lojik 0 ise veri yolu 4 bit

Komutların Ayrıntılı Açıklamaları

Display Temizle

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Şekil 1.25: Display Temizle Yazmacı

Tüm ekranı temizler ve kursörü 0. adres olan başlangıç adresine getirir. Display Data RAM'deki tüm adreslere boşluk koduna karşılık gelen 20H yazılır. DDRAM için adres sayacı 0 yapılır. Eğer kaydırılmışsa displayi orijinal konumuna getirir. Herhangi bir karakter displayde görülmez ve imleç displayin başına gelir. Komut uygulama zamanı: 82µs-1.64ms (8bit modda), 120µs- 4.9ms (4bit modda)

İmleç Başa Komutu

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	X

Şekil 1.26: İmleç Başa Yazmacı

Kursörü displayin başına getirir. DDRAM için adres sayacını 0 yapar. DDRAM içeriğini etkilemez. X etkisiz giriş anlamındadır. Display eğer kaydırılmış ise original durumuna getirir. Bu işlem için LCD ye 0x02 veya 0x03 komutu gönderilir. Komut uygulama zamanı: 40µs-1.64 ms (8bit modda), 120µs-4.8 ms (4bit modda)

Giriş Modu Komutları

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

Şekil 1.27: Giriş Modu Yazmacı

DDRAM adreslerinin artışına ya da azalışına göre kursör hareket yönünü ayarlar. Bu çalışmalar veri yazma ve okuma sırasında icra edilir. Bir karakter kodu DDRAM'e yazıldığıda ya da karakter kodu DDRAM'den okunduğunda; I/D=1 olduğunda DDRAM adresi 1 arttırılır ve kursör sağa hareket eder. I/D=0 olduğunda DDRAM adresi 1 azaltılır ve kursör sola hareket eder. Yukarıda bahsedilen çalışma CGRAM'den okuma ve CGRAM'e yazma sırasında da geçerlidir.

S=1 yapıldığında kursörün sağa hareketi sırasında, displayi karakter sayısı kadar komple sola kaydırır, 0 ise kursörün sağa ya da sola kaymasına izin verilir. Bu anlatılanlar şekil de tablo halinde özetlenmiştir. Tablodan DB2 biti daima lojik 1 dir. Açıklaması yapılan her komutun 2'li ve 16 sistemde karşılıkları verilmiştir. Örneğin LCD de gösterilen her bir karakterden sonra imlecin bir sağa kaymasını istiyorsak 16 sayı sisteminde 0x 6 h veya ikili sayı sisteminde 110 komutunu kullanmamız gerekir.

DB2	DB1	DB0		
1	I/D	S	**Açıklama**	
1	0	0	0x 4	LCD de gösterilen her bir karakterden sonra imleç bir sola kayar.
1	0	1	0x 5	LCD de gösterilen her bir karakterden sonra imleç bir sola kayar, displayde bir sola kayar.
1	1	0	0x 6	LCD de gösterilen her bir karakterden sonra imleç bir sağa kayar.
1	1	1	0x 7	LCD de gösterilen her bir karakterden sonra imleç bir sağa kayar, displayde bir sağa kayar.

Şekil 1.28: Giriş Modu Komut Tablosu

Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

Display, İmleç Aç/Kapa

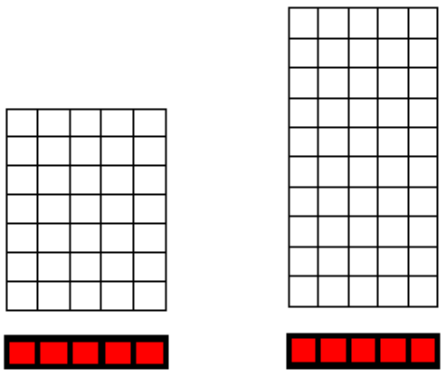
RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

Şekil 1.29 : Display, İmleç Aç/Kapa Yazmacı

Displayi ve kursörü açıp kapatır, kursörün görünümünü ayarlar.

D=1 ise display açılır. D=0 ise display kapanır. Kapalıyken display verisi enerji olduğu sürece DDRAM'de saklanır. C=1 ise kursör ekranda görüntülenir. C=0 ise kursör ekranda görüntülenmez.

B=1 ise kursör ile gösterilen tüm noktalar yanıp söner. B=0 ise sadece kursör ekranda görülür. Kursör 5x7 dot karakterde 8. satırda, 5x10 dot karakterde 11. satırda görüntülenir. Aşağıdaki Şekil 1.30'da en alt satır kursör için ayrılmıştır. Şekil 1.31'de komutların özeti tablo halinde verilmiştir.



Şekil 1.30: 5x7, 5x10 Karakter LCD

1	D	C	B	Açıklama	
1	0	0	0	0x 8	Display kapalı, imleç gösterilmez, imleç yanıp sönmez
1	0	0	1	0x 9	Display kapalı, imleç gösterilmez, imleç yanıp söner
1	0	1	0	0x 0A	Display kapalı, imleç gösterilir, imleç yanıp sönmez
1	0	1	1	0x 0B	Display kapalı, imleç gösterilir, imleç yanıp söner
1	1	0	0	0x 0C	Display açık, imleç gösterilmez, imleç yanıp sönmez
1	1	0	1	0x 0D	Display açık, imleç gösterilmez, imleç yanıp söner
1	1	1	0	0x 0E	Display açık, imleç gösterilir, imleç yanıp sönmez
1	1	1	1	0x 0F	Display açık, imleç gösterilir, imleç yanıp söner

Şekil 1.31: Display, İmleç Aç/Kapa Komut Tablosu

Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

Kursör ve Display Kaydır

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	x	x

Şekil 1.32:Kursör ve Display Kaydırma Yazmacı

S/C (**S**hift/ **C**ursor) : Lojik 1 ise display kaydırılır, Lojik 0 ise imleç kaydırılır.

R/L (**R**ight/**L**eft:ojik 1 ise sağa kaydırılır, Lojik 0 ise sola kaydırılır.

DDRAM içeriğini değiştirmeden kursorü hareket ettirir ve displayi sağa veya sola kaydırır. 2 satırlı displayde kursör 1. satırın 40. sütununu geçtikten sonra 2. satıra geçer. 1. ve 2. satırlar aynı anda kayar. Bu anlatılanlar şekil de tablo halinde özetlenmiştir.

DB4	DB3	DB2	DB1	DB0		
1	S/C	R/L	x	x	**Açıklama**	
1	0	0	x	x	0x10	Kursör sola kayar. Adres sayacı 1 azalır
1	0	1	x	x	0x 14	Kursör sağa kayar. Adres sayacı 1 artar.
1	1	0	x	x	0x18	Tüm display sola kayar. Kursör display ile birlikte kayar.
1	1	1	x	x	0x 1C	Tüm display sağa kayar. Kursör display ile birlikte kayar

Şekil 1.33: Kursör ve Display Kaydırma Komut Tablosu

Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

Fonksiyon Ayar Komutları

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	x	x

Şekil 1.34:Fonksiyon Ayar Yazmacı

Bu komut grubu veri uzunluğunu, display satır sayısını ve karakter fontunu ayarlar. Meşgul bayrağı ve adres okumanın dışında herhangi bir komut uygulamadan önce programın başında fonksiyon kurulumu yerine getirilmelidir

DL: komutu veri uzunluğunu ayarlar. DL=1 ise gönderilen veya alınan veri 8 bit uzunluğundadır (DB7-DB0). DL=0 ise gönderilen veya alınan veri 4 bit uzunluğundadır (DB7-DB4). 4 bit veri uzunluğu seçilirse verinin önce üst dört biti sonra da alt dört biti gönderilmelidir.

N: Display satır sayısını belirtir. Lojik 1 ise 2 satır kullanılır, Lojik 0 ise 1 satır.

F: Karakter fontunu ayarlar. Lojik 1 ise 5x10 karakter fontu, Lojik 0 ise 5x7 karakter fontu kullanılır.

DB5	DB4	DB3	DB2	DB1	DB0		
1	**DL**	**N**	**F**	**x**	**x**	**Açıklama**	
1	1	1	1	x	x	0x3C	8 bit iletişim, LCD 2 satır, 5x10 dot matris,
1	1	1	0	x	x	0x 38	8 bit iletişim, LCD 2 satır, 5x7 dot matris,
1	1	0	1	x	x	0x34	8 bit iletişim, LCD 1 satır, 5x10 dot matris,
1	1	0	0	x	x	0x 30	8 bit iletişim, LCD 1 satır, 5x7 dot matris,
1	0	1	1	x	x	0x2C	4 bit iletişim, LCD 2 satır, 5x10 dot matris,
1	0	1	0	x	x	0x28	4 bit iletişim, LCD 2 satır, 5x7 dot matris,
1	0	0	1	x	x	0x24	4 bit iletişim, LCD 1 satır, 5x10 dot matris,
1	0	0	0	x	x	0x20	4 bit iletişim, LCD 1 satır, 5x7 dot matris,

Şekil 1.35:Fonksiyon Ayar Komutları Tablosu

5x10 noktalı karakter fontu 2 satır displayde kullanılmaz. 5x10 noktayı LCD'nin desteklemesi gereklidir. Desteklemiyorsa 5x7 nokta ayarlanmalıdır. Komut uygulama zamanı: 40µs (8bit modda) 120µs (4bit modda).

CGRAM Adresi Kur

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Şekil 1.36:CGRAM Adresi Kurma Yazmacı

CGRAM adresini kurar. CGRAM verisi bu ayarlamadan sonra gönderilir. Adres sayacında 6 bitlik CGRAM adresi oluşur. Veri bundan sonra CGRAM için mikroişlemciden okunur veya mikroişlemciye yazılır.

Komut uygulama zamanı: 40µs (8bit modda),120µs (4bit modda)

DDRAM Adresi Kur

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Şekil 1.37 : DDRAM Adresi Kurma Yazmacı

DDRAM adresini kurar. DDRAM verisi bu ayarlamadan sonra gönderilir. Adres sayacında 6 bitlik DDRAM adresi oluşur. Veri bundan sonra DDRAM için mikroişlemciden okunur veya mikroişlemciye yazılır.

1 Satırlı LCD'de DDRAM adres aralığı 00H - 4FH arasında iken, 2x16 Satırlı LCD'de

1. satır için DDRAM adres aralığı 00 h – 0F h dir.
2. satır için DDRAM adres aralığı 40 h – 4Fh dır.

2x16 Satırlı LCD'de mikrodenetleyiciden adres kurma komutu gönderirken DB7 biti lojik 1 olduğu için adres kurma komutu

1.satır için 80H ile 8FH arasındadır.
2.satır için bu aralık C0 H ile CF H arasındadır.

LCD'nin 1.Satır ve 8. Sütununu kurmak için 1. Satırın ilk sütununun adres kurma komutuna (80H) sütun sayısının 1 eksiği (8-1) ilave edilir. Şekil 1.38' de görüldüğü gibi ikili olarak 10000111 sayısının veya 16 sayı sisteminde 87 H komutunun gönderilmesi gerekir.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0
0	0	1	0	0	0	0	1	1	1

Şekil 1.38: LCD nin 1. Satır ve 8. Sütununu Kurma Komutu

Yine benzer şekilde LCD nin 2. Satır ve 5. Sütunu kurmak için 2. Satırın ilk sütununun kurma komutuna (C0h) sütun sayısının 1 eksiği (5-1) ilave edilir. Şekil de görüldüğü gibi ikili olarak 10000100 sayısının veya 16 sayı sisteminde C4 h komutunun gönderilmesi gerekir.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0
0	0	1	1	0	0	0	1	0	0

Şekil 1.39 : LCD nin 2. Satır ve 5. Sütununu Kurma Komutu

Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

Meşgul Bayrağı ve Adresi Oku

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Şekil 1.40 : Meşgul Bayrağı ve Adres Okuma Yazmacı

LCD içerisinde bir çalışmanın olduğunu ifade eden meşgul bayrağını okur. Diğer görevi de adres sayacının içeriğini okumaktır. BF=1 okunuyorsa LCD'nin meşgul olduğu anlaşılır. BF=0 olmadan yeni bir komut LCD tarafından kabul edilmez. BF durumu bir sonraki komut gönderilmeden önce kontrol edilmelidir.

Aynı zamanda 7 bitlik adres sayacı değeri bu komut ile okunur. Adres sayacı hem CGRAM hem de DDRAM adresleri için kullanılır. O andaki kullanımı bir önceki komuta bağlıdır. Komut uygulama zamanı: 0-1µs (Her iki modda)

CG/DD RAM Adresine Veri Yaz

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D7	D6	D5	D4	D3	D2	D1	D0

Şekil 1.41 : CG/DD RAM Adresine Veri Kayıt Yazmacı

DDRAM veya CGRAM'a 8 bitlik veri yazar. CGRAM veya DDRAM ile ilgili bu işlemler daha önce yapılan tanımlama ve adres kurma ile belirlenir. Örneğin yazma işleminden sonra adresin 1 artması veya azalması giriş kur komutuna gore gerçekleşir. Giriş kurma aynı zamanda display kaymasını da belirler. Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

CG/DD RAM'den Veri Oku

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D7	D6	D5	D4	D3	D2	D1	D0

Şekil 1.42 : CG/DD RAM'den Veri Okuma Yazmacı

CGRAM veya DDRAM'dan 8 bitlik formatta veri okur. Daha önceki komut uygulamaları ile CGRAM'ın ya da DDRAM'ın okunacağı belirlenir. Okuma komutuna başlamadan önce ya CGRAM ya da DDRAM için adres kurma komutu uygulanmalıdır. Eğer bu yapılmazsa ilk okuma verisi iptal olacaktır. Okumadan sonra giriş moduna bağlı olarak adres otomatik olarak 1 arttırılır veya azaltılır.

Komut uygulama zamanı: 40µs (8bit modda), 120µs (4bit modda)

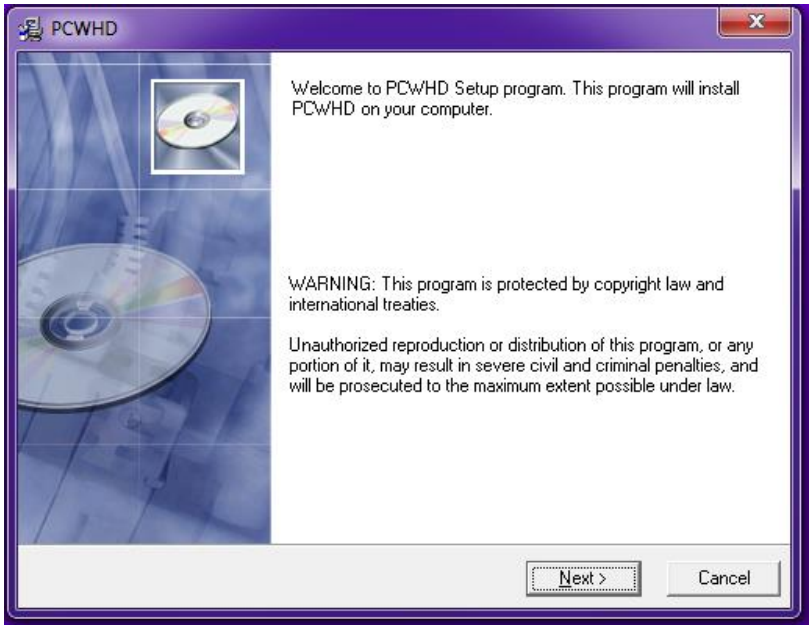
1.3. LCD Programlanması

1.3.1. PIC C ile LCD Kodu Yazımı ve Proteus Isıs ile Simülasyonu

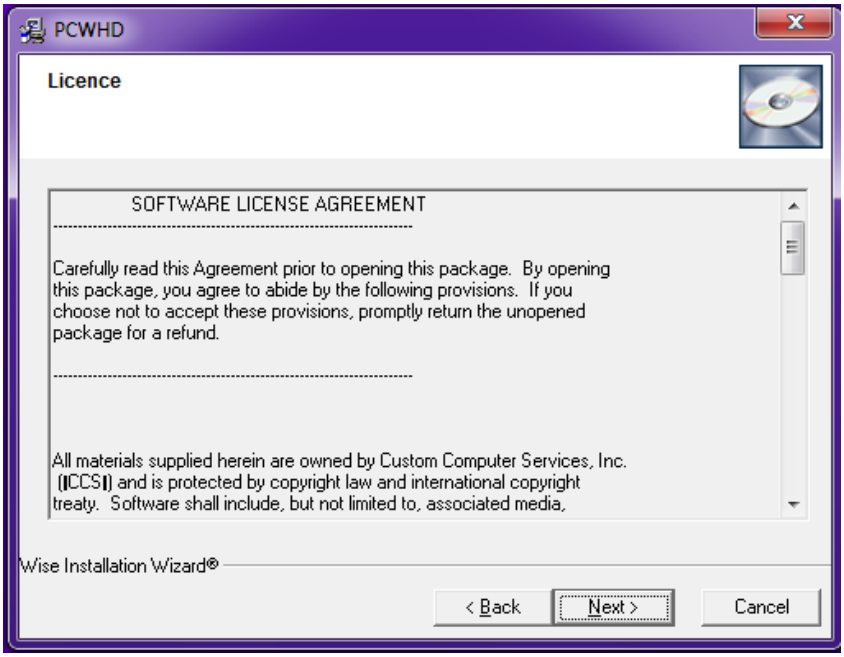
1.3.1.1. Ccs C Demo Programının Kurulması

http://www.ccsinfo.com adresinden CCS C derleyicisi satın alınabilir veya demo sürümü indirilebilir. Demo sürümü belirli bir süre için çalışmakta ve çalışması için internet bağlantısı gerektirmektedir.

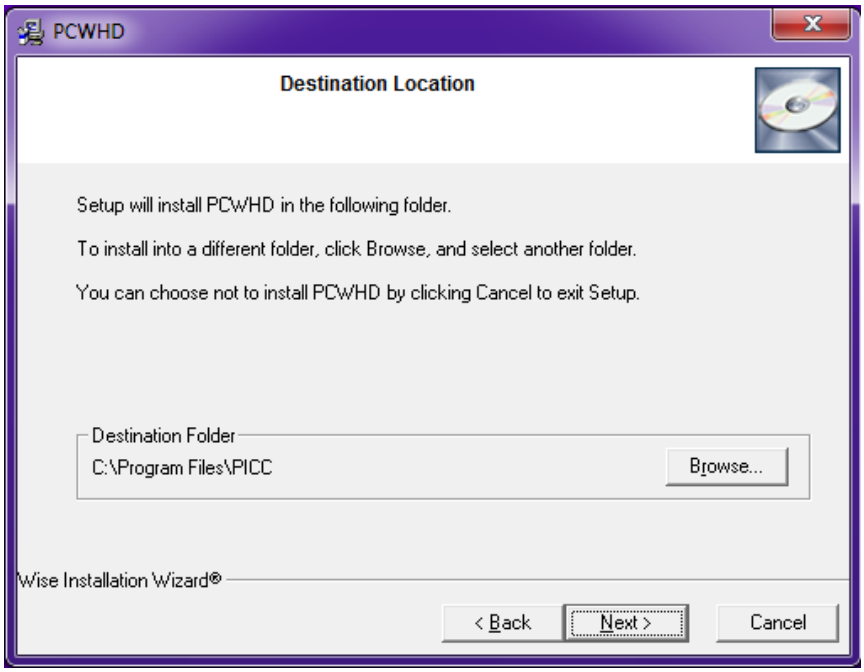
Kurulum başlangıcında Şekil 1.43 teki ekran gelir. Next butonuna basınca önümüze Şekil 1.44 teki ekran gelir. Tekrar Next botonuna basıldığında programın kurulacağı dizini gösteren veya isteğe bağlı olarak dizini değiştirmeye imkan sağlayan Şekil 1.45 teki ekran gelir. Next botonuna basıldıktan sonra Şekil 1.46 daki Windows sistemindeki grup ismi seçim ekranı gelir. Tekar Next botonuna basıldıktan sonra Şekil 1.47 deki kuruluma başlama penceresi gelir. Next butonuna basıldıktan sonra kurulum tamamlanır.



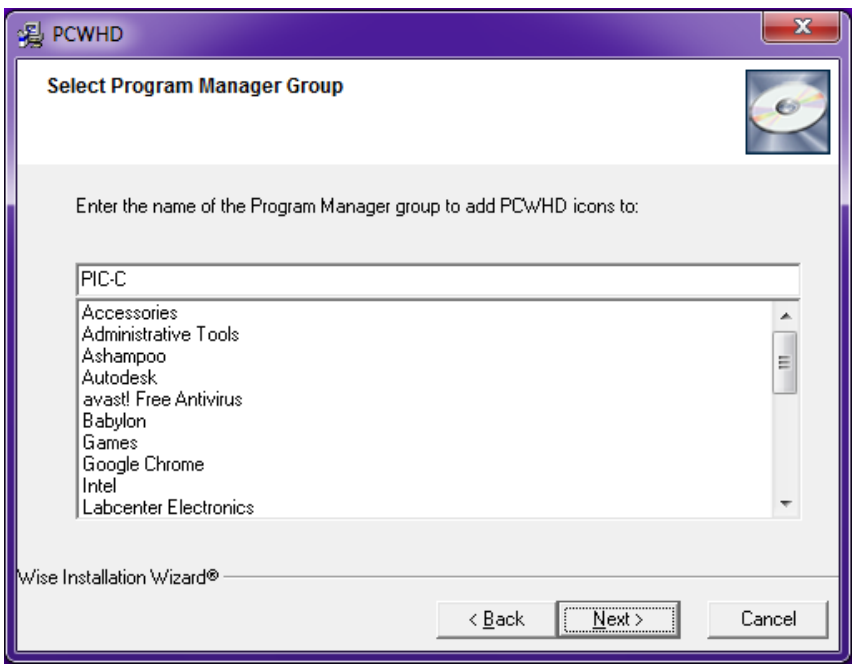
Şekil 1.43: CCS C Demo Programı Kurulum Başlangıcı



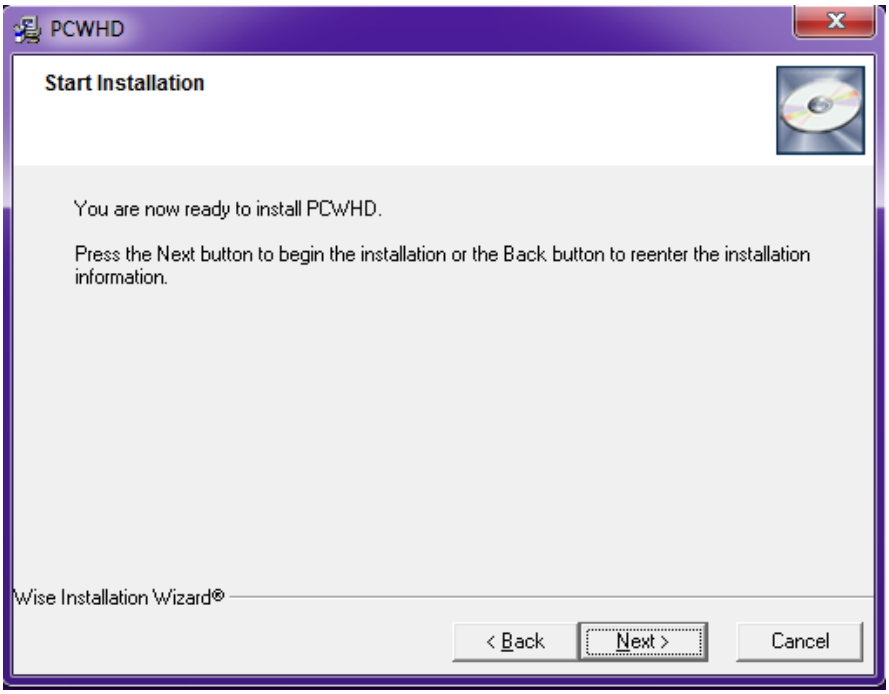
Şekil 1.44: Yazılım Lisans Anlaşması



Şekil 1.45: Program Kurulum Yeri Seçimi



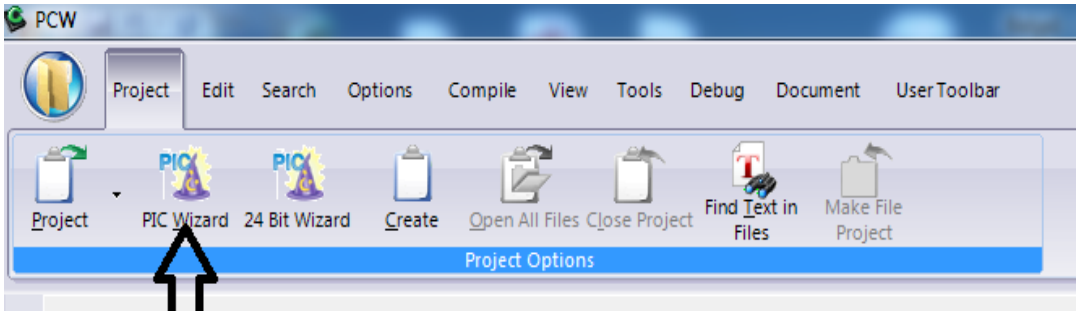
Şekil 1.46: Grup İsmi Seçim Ekranı



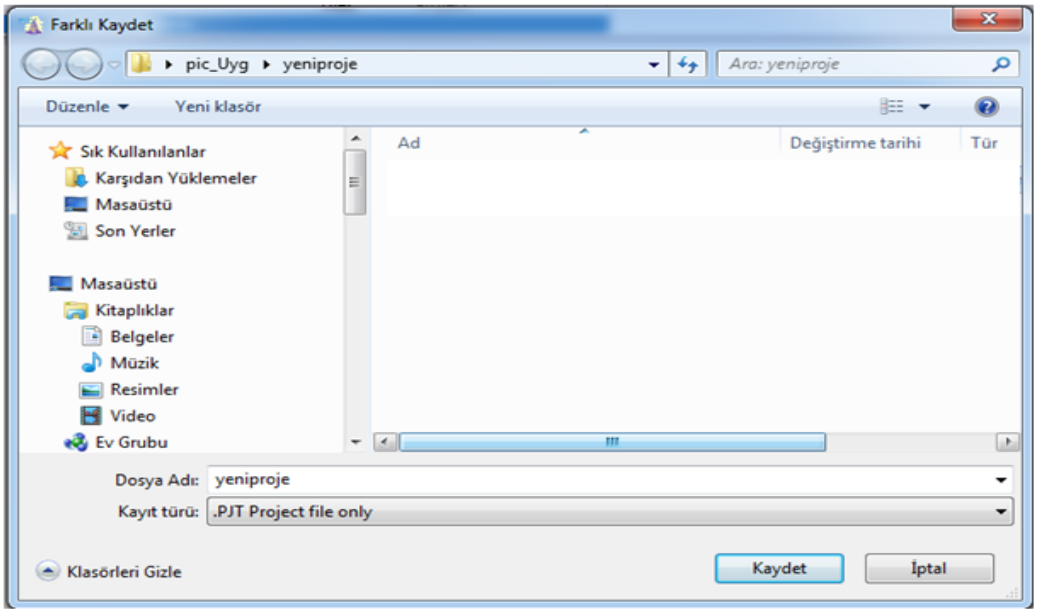
Şekil 1.47: Kuruluma Başlama Ekranı

1.3.1.2. Otomatik Olarak Proje Dosyası Oluşturmak ve Derlemek

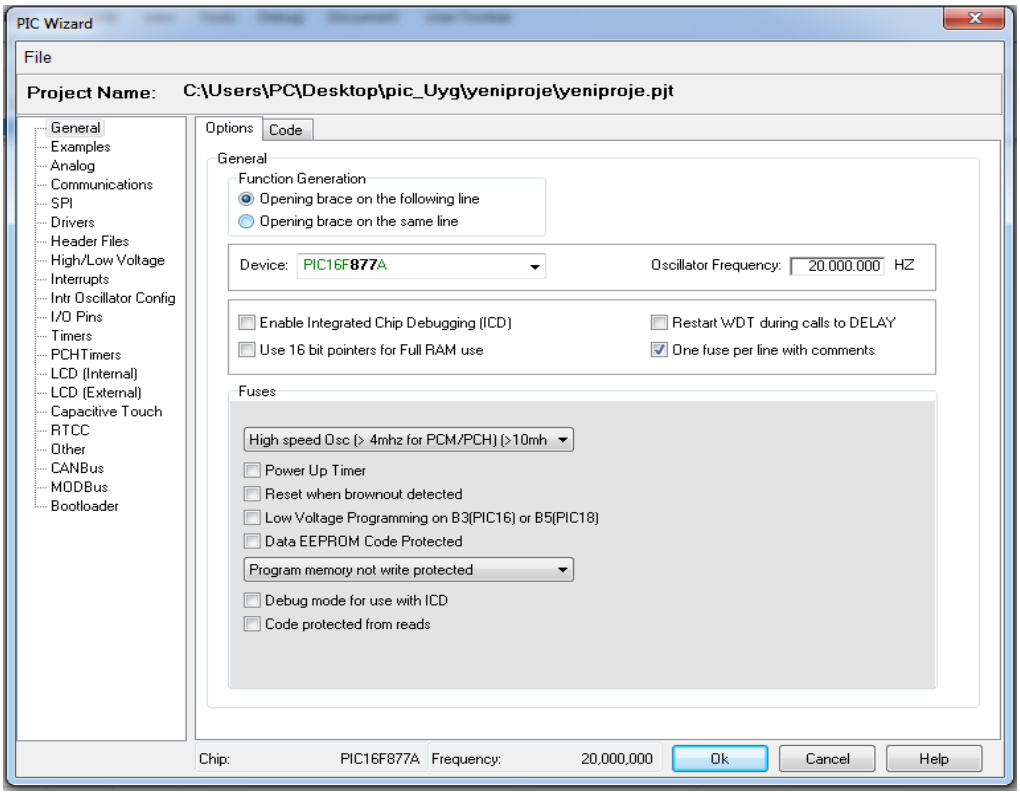
CCS C programını başlatmak için masaüstü veya başlat menüsünden sembolüne tıklanır. Otomatik olarak proje dosyası oluşturmak için Şekil 1.48 de gösterilen Project menüsünden PIC_Wizard düğmesine tıklanır. Şekil 1.49 daki pencerede dosyanın kaydedileceği yer ismi belirtilir. Kaydetme işleminden sonra Şekil 1.50 deki proje dosyası ile ayarlar menüsü ekrana gelir. Burada istenen sekme tıklanarak programda kullanılmak istenen port giriş çıkış durumları, zamanlayıcılar, adc, seri iletişim v.b ile ilgili seçim ve ayarların kod karşılıklarının programa otomatik olarak geçmesi sağlanır. Bu modüldeki uygulamalarımızda kodun tamamı verilmiştir. Bu yüzden şekil 1.50 deki pencerede doğrudan OK düğmesine basılabilir. Daha sonra yaptığımız seçimlerle bağlı olarak otomatik olarak üretilen C kodu Şekil 1.51 deki gibi ekrana gelir. Bu modülde uygulayacağınız C kodlarını bu alana yazabilir veya yapıştırabilirsiniz, bunu yapmadan önce otomatik olarak üretilen C kodunu siliniz. Derleme işlemi için Şekil 1.52 deki pencerede Compile menüsüne, buradan Compile sekmesine tıklanır. Derleme işlemi başarılı ise birkaç saniye süreyle Şekil 1.53 teki pencere ekrana gelir.



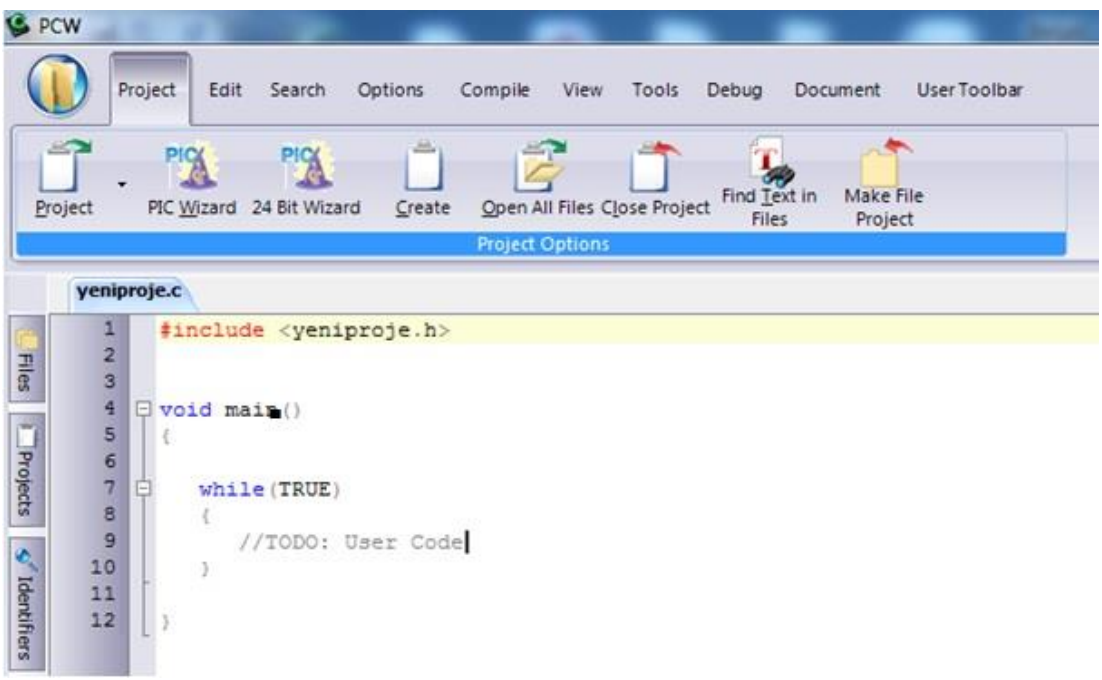
Şekil 1.48: Project Menüsü İçeriği



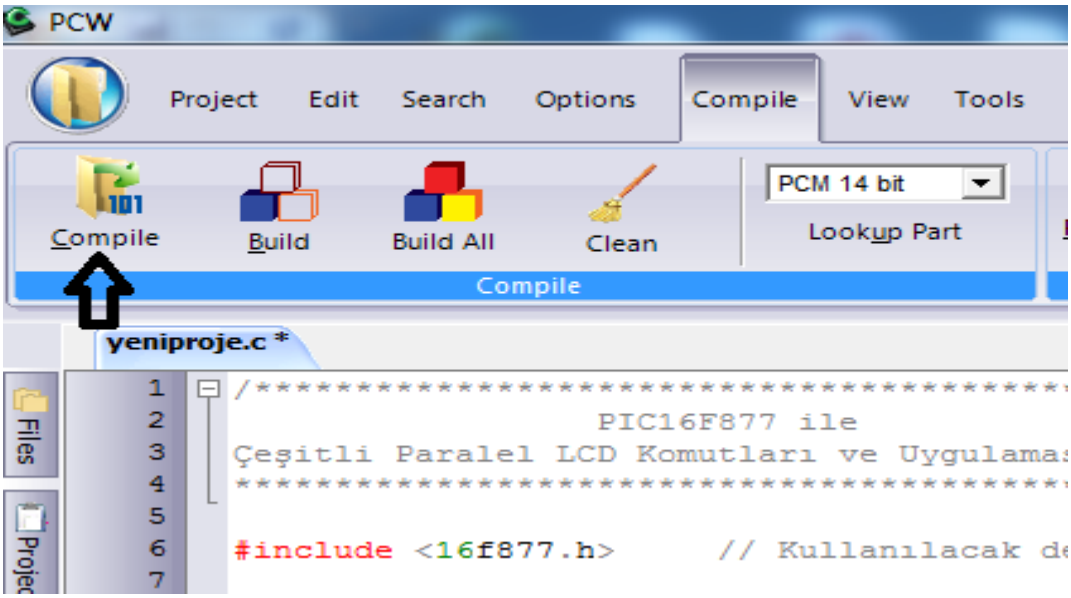
Şekil 1.49: Proje Kayıt Penceresi



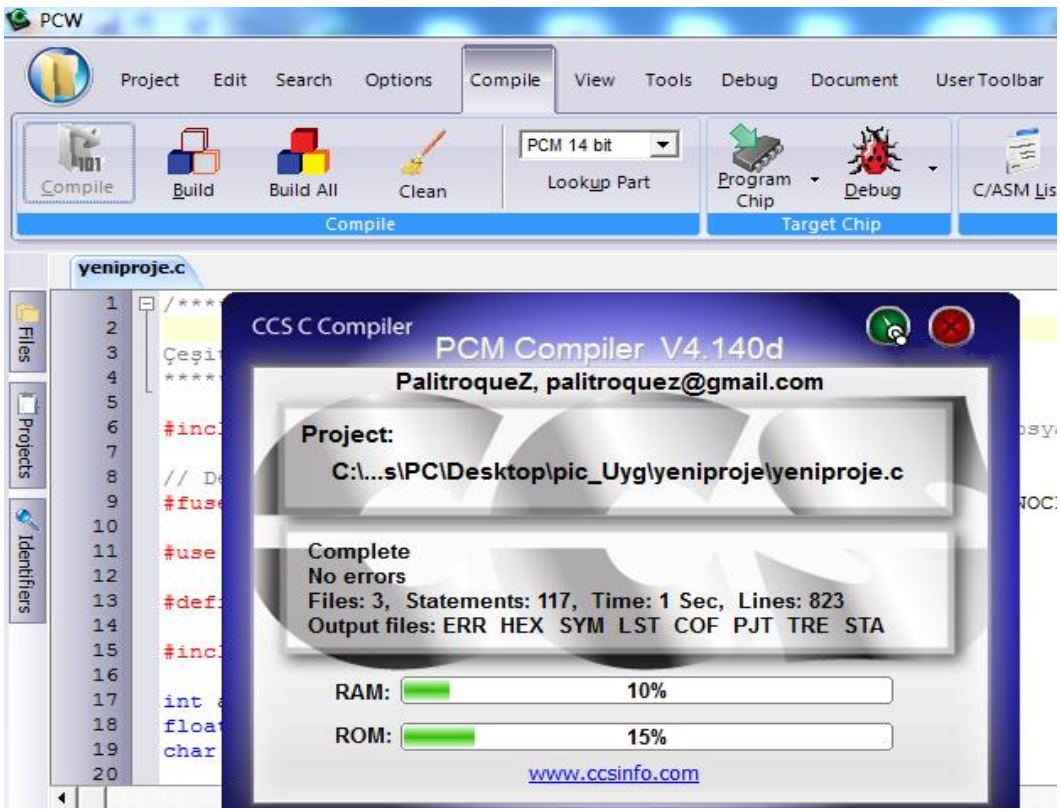
Şekil 1.50: Proje Ayar Penceresi



Şekil 1.51: Şekil 1.50 deki Seçimlere Bağlı Olarak Otomatik Üretilen Kod



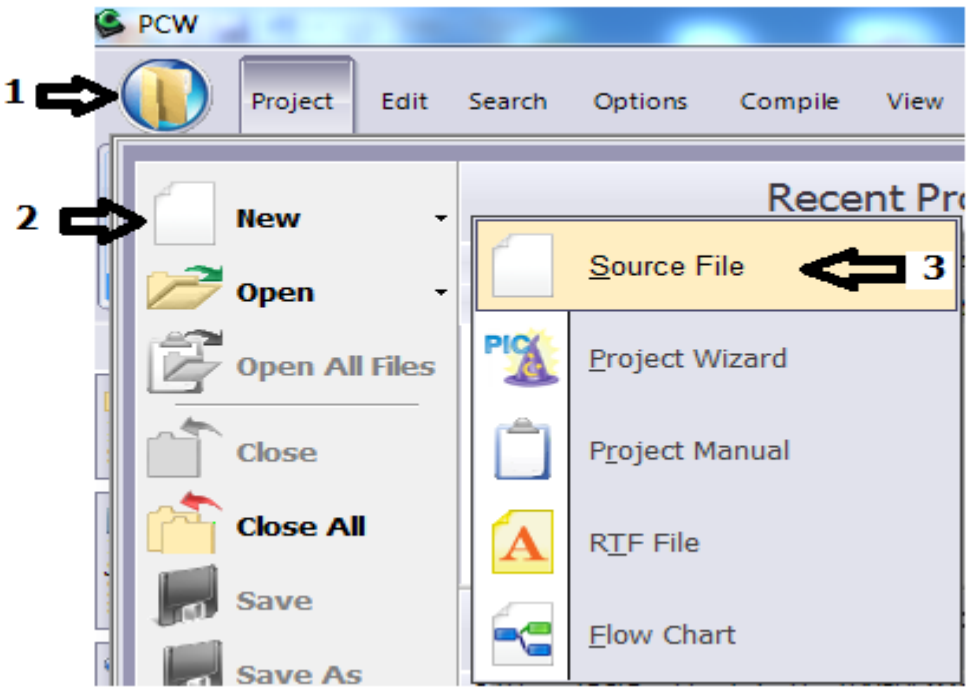
Şekil 1.52: Derleme Ekranı



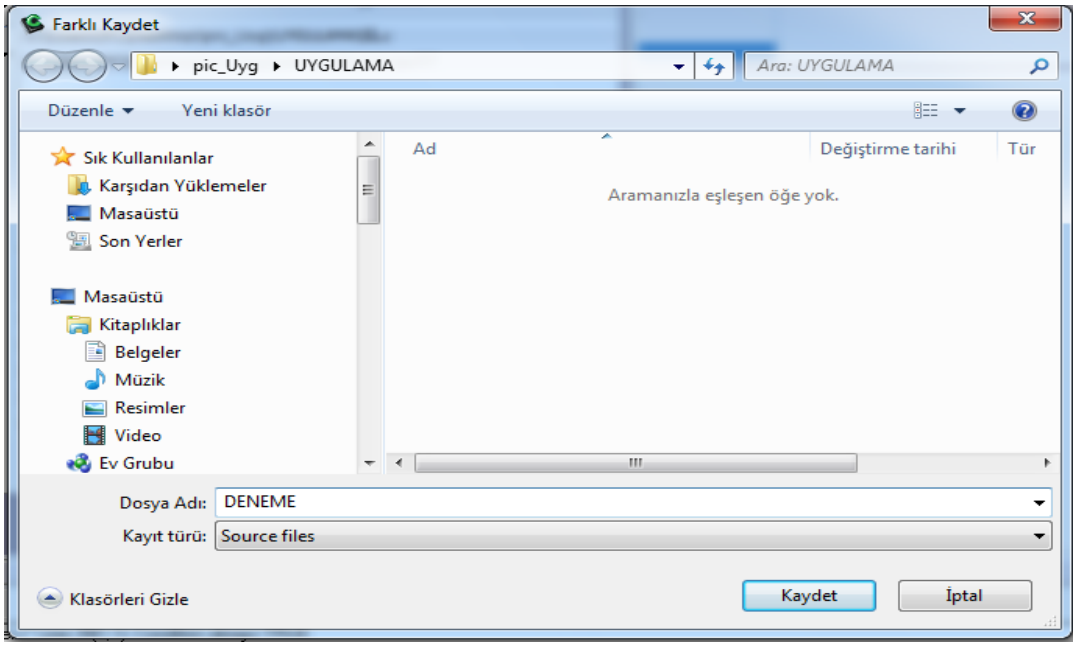
Şekil 1.53: Derleme Başarılı Ekranı

1.3.1.3. El İle (Manuel Olarak) Proje Dosyası Oluşturmak ve Derlemek

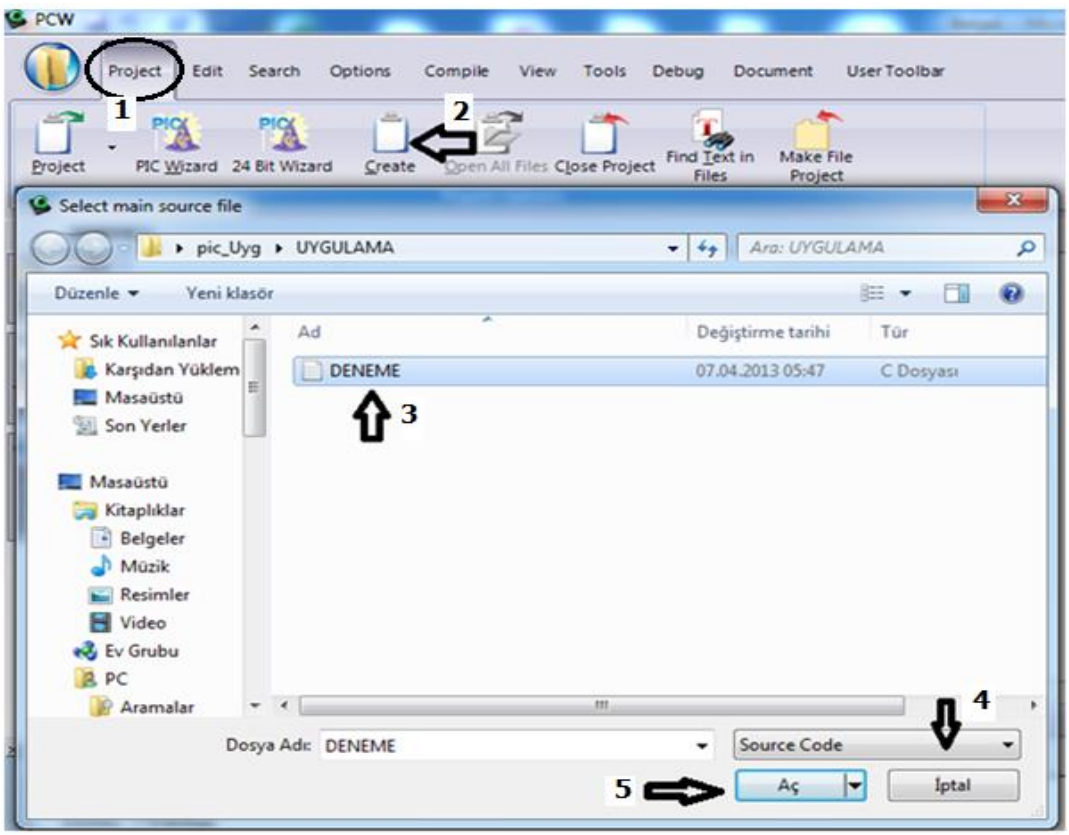
Şekil 1.54'te görüldüğü gibi 1. adımda dosya sembolüne tıklanır. 2. adımda New komutuna tıklanır. 3. adımda ise Source File komutuna tıklanır. Şekil 1.55 te görüldüğü gibi dosyanın kayıt yeri ve ismi belirtilirek kayıt yapılır. Şekil 1.56 da görüldüğü 1. adımda gibi Project menüsüne tıklanır ve 2. adımda Create komutu tıklanır, 3. adımda açılır menüden C uzantılı kaynak dosya seçilir, 4. adımda dosya türü olarak source code seçili olmalıdır, 5. adımda aç butonuna tıklanır. Şekil 1.57 de Project Option menüsü ile kullanılacak mikrodenetleyici, kaynak dosya (source file) seçilir Apply butonuna basılır. Bu modüldeki uygulamalarımızda kodun tamamı verilmiştir. Bu yüzden şekil 1.57 deki pencerede hiç bir şey yapmadan doğrudan Apply düğmesine basılabilir. Daha sonra yaptığımız seçimlerle bağlı olarak otomatik olarak üretilen C kodu Şekil 1.51 deki gibi ekrana gelir. Bu modülde uygulayacağınız C kodlarını bu alana yazabilir veya yapıştırabilirsiniz, bunu yapmadan önce otomatik olarak üretilen C kodunu siliniz. Derleme işlemi için Şekil 1.52 deki pencerede Compile menüsüne, buradan Compile sekmesine tıklanır. Derleme işlemi başarılı ise şekil 1.53 deki pencere ekrana gelir.



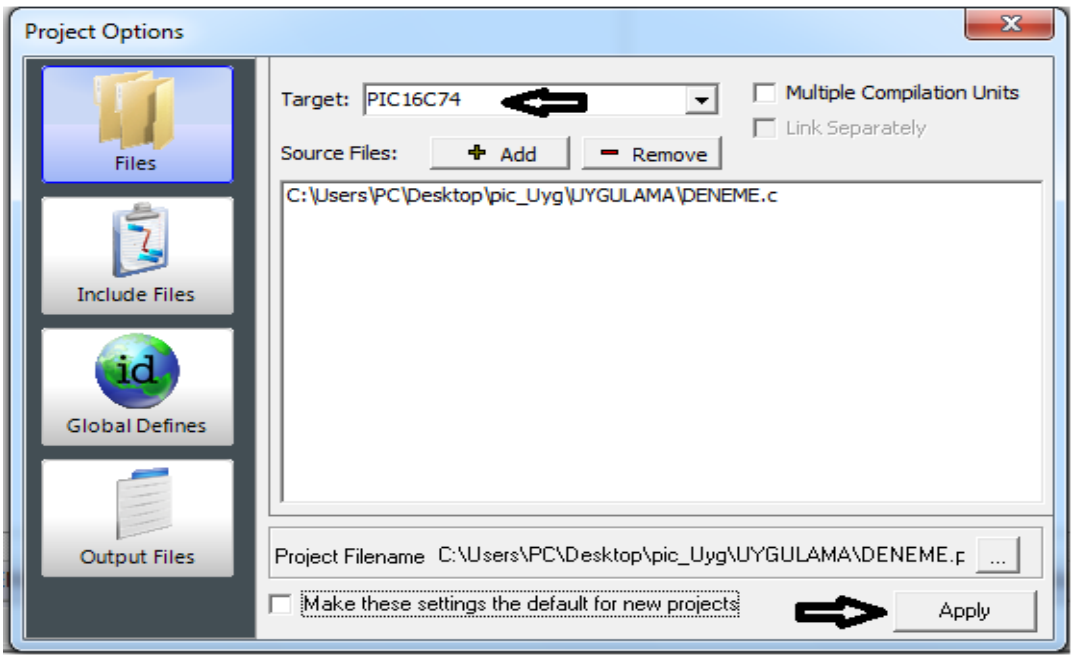
Şekil 1.54: Project Menüsü



Şekil 1.55: Dosya Kayıt Yeri ve İsmi Seçimi



Şekil 1.56: Mevcut Dosyayı Seçerek Proje Oluşturma Ekranı



Şekil 1.57: Proje Seçenekleri Penceresi

1.3.1.4. PIC C Programı ve LCD ile İlgili Özellikleri

CCS C derleyici bilgisayara yüklendikten sonra istenirse MPLAB programına bağlanabilir. Bilgisayardaki PIC C klasörü incelendiğinde iki alt klasör göze çarpar. Bunlar "Devices" ve "Drivers" klasörleridir. "Devices" klasörü içerisinde mikrodenetleyicilerin donanım tanımlarının yer aldığı başlık dosyaları yer almaktadır. Bu dosyaların uzantısı .h ile biter. Programda PIC16F877 entegresi ile ilgili donanım kullanılacağından dolayı "#include <16F877.h>" komutu programın en başında kullanılmalıdır. Aslında 16F877.h dosyası C dili ile yazılmıştır. İncelendiğinde PIC16F877 mikrodenetleyicisinin donanım kullanımına yönelik ifadeler ve fonksiyonlar göze çarpar. 16F877 başlık dosyası programın başında tanımlandığında başlık dosyasında tanımlanan fonksiyonlar ve ifadeler yazılan PIC-C programında kullanılabilir. 16F877.h başlık dosyasını inceleyiniz. "Drivers" klasörü içerisinde çeşitli elektronik yapıların sürücü dosyaları yer almaktadır. Burada yer alan dosyaların uzantıları ".c" ile biter. Bu klasörde LCD'ye özel LCD.C dosyası vardır ve bu dosya içerisinde LCD kulanımı için fonksiyonlar yer almaktadır. Eğer 2 satır LCD kullanılmak isteniyorsa LCD.C sürücü dosyası programın başında tanımlanmalıdır. Bu amaçla "#include <LCD.C>" komutu kullanılır. Eğer 4 satır LCD kullanılmak isteniyorsa programın başında "#include <LCD420.C>" kullanılır. Aşağıda CCS C derleyicisi ilk kurulduğunda LCD.C dosyasının içeriğinden bir kısım görülmektedir. Aşağıda görüldüğü gibi, sürücünün en başında sürücü tanımlandığında kullanılabilecek fonksiyonlar gösterilmiştir.

LCD.C sürücü dosyasını inceleyiniz.

```
////////////////////////////////////////////////////////////////////////
////                        LCDD.C                                  ////
////              Driver for common LCD modules                     ////
////                                                                ////
////  lcd_init()   Must be called before any other function.        ////
////                                                                ////
////  lcd_putc(c)  Will display c on the next position of the LCD.  ////
////                The following have special meaning:             ////
////                 \f  Clear display                              ////
////                 \n  Go to start of second line                ////
////                 \b  Move back one position                     ////
////                                                                ////
////  lcd_gotoxy(x,y) Set write position on LCD (upper left is 1,1)  ////
////                                                                ////
////  lcd_getc(x,y)   Returns character at position x,y on LCD      ////
////                                                                ////
// As defined in the following structure the pin connection is as follows:
//     D0  enable
//     D1  rs
//     D2  rw
//     D4  D4
//     D5  D5
//     D6  D6
//     D7  D7
//
//   LCD pins D0-D3 are not used and PIC D3 is not used.

// Un-comment the following define to use port B
// #define use_portb_lcd TRUE
struct lcd_pin_map {                // This structure is overlayed
BOOLEAN enable;           // on to an I/O port to gain
BOOLEAN rs;               // access to the LCD pins.
BOOLEAN rw;               // The bits are allocated from
BOOLEAN unused;           // low order up.  ENABLE will
int     data : 4;         // be pin B0.
} lcd;
```

LCD.C sürücüsü HD44780 temelli tüm LCD'ler için kullanılabilen bir dosyadır. İlk başlangıç değeri 4 bitlik çalışma modu için uygundur. Fakat bu haliyle programda kullanılmaz. Öncelikle kullanılacak donanıma uyup uymadığı incelenmelidir. Bu haliyle LCD'nin Port D'ye bağlanması gereklidir. Mikrodenetleyicinin hangi pinlerinin LCD ye bağlanabileceği konusunda eski ve yeni versiyon PIC C progamlarında bazı farklılıklar bulunmaktadır. Aşağıda bunlardan bahsedilecektir.

Eski ve Yeni Tüm PIC C Versiyonlarında Mikrodenetleyici ve LCD bağlantıları

LCD B veya D portuna bağlanabilir, bağlantıların aşağıda şekil de gösterildiği gibi yapılması gerekir.

LCD B portuna bağlı ise	
Mikrodenetleyici Pin Adı	LCD Pin Adı
B0	E
B1	R/S
B2	R/W
B4	D4
B5	D5
B6	D6
B7	D7

LCD D portuna bağlı ise	
Mikrodenetleyici Pin Adı	LCD Pin Adı
D0	E
D1	R/S
D2	R/W
D4	D4
D5	D5
D6	D6
D7	D7

Şekil 1.58: Mikrodenetleyici ve LCD bağlantıları

Eğer programda Mikrodenetleyicinin hangi portunun LCD'ye bağlandığı belirtilmez ise PIC C programı D portuna bağlı olduğunu kabul eder. Eğer LCD B portuna bağlandı ise #include <lcd.c> komutundan önce #define use_portb_lcd TRUE komutunun kullanılması gerekir.

lcd.c dosyası içerisinde gecikme fonksiyonları kullanıldığından gecikme fonksiyonlarının kullanılacağı osilatör frekansını belirten #use delay () komutu mutlaka #include <lcd.c> komutundan önce kullanılmalıdır. Örneğin osilatör frekansı 4MHz ise komut #use delay (clock=4000000) şeklinde yazılmalıdır.

Yeni PIC C Versiyonlarında Mikrodenetleyici ve LCD bağlantıları

Yeni versiyonlarda LCD nin sadece B veya D portlarına bağlanma kısıtı ortadan kaldırılmıştır. Yeni versiyon CCS C de iki yöntem bulunmaktadır.

Yöntem 1 : Bu yöntemde LCD doğrudan B veya D portuna bağlanır. B veya D portu için #include <lcd.c> komutundan önce aşağıdaki tanımların yapılması gerekir.

#define LCD_DATA_PORT getenv ("SFR:PORTB") // LCD B potuna bağlı ise

#define LCD_DATA_PORT getenv ("SFR:PORTD") // LCD D potuna bağlı ise

Yöntem 2 : Bu yöntemde mikrodenetleyicinin istenilen Portunun istenilen pini LCD ye bağlanabilir. Bunun için #include <lcd.c> komutundan önce aşağıdaki örnekte verilen tanımlara benzer LCD nin hangi pinin denetleyicinin hangi pinine bağlı olduğunu belirten yapıların oluşturulması gerekir.

```
#define LCD_ENABLE_PIN  PIN_B0
#define LCD_RS_PIN   PIN_E2
#define LCD_RW_PIN  PIN_E1
#define LCD_DATA4  PIN_ A3
#define LCD_DATA5  PIN_B7
#define LCD_DATA6   PIN_C2
#define LCD_DATA7  PIN_ B3
```

Bu modülde genele hitap etmek için eski ve yeni versiyonlarda kullanılabilen metot tercih edilecektir.

LCD.C ve LCD420.C Sürücüsü Dosyalarında Bulunan Hazır Fonksiyonlar

1. lcd_init()

LCD'yi hazırlamak için mutlaka diğer LCD fonksiyonlarından önce tanımlanmalıdır.

2. lcd_putc(c)

LCD ekranında istenilen koordinata "c" ile ifade edilen karakteri veya karakter dizisini yazar. Tek karakter gönderilecek ise tek tırnak içine 'K' alınır, eğer bir string (karakter dizisi) gönderilecek ise çift tırnak içine "EML" alınır. Bu fonksiyon ile birlikte tırnak içinde yardımcı anahtarlar kullanılabilir.

\f : Ekranı temizler ve kursörü 1. satırın 1.sütununa alır.
\n : Kursörü 2. satırın başına getirir.
\b : Kursörü 1 adım geriye hareket ettirir.

3. lcd_gotoxy(x,y)

LCD ekranında yazma koordinatını ayarlar. X sütun numarasını, Y ise satır numarasını temsil eder. LCD ekranında sol üst köşenin koordinatı (1,1)'dir.

4. lcd_getc(x,y)

(x,y) ile ifade edilen koordinattaki karakteri fonksiyon üzerinden döndürür. Diğer bir ifade ile istenen koordinattaki karakteri okur.

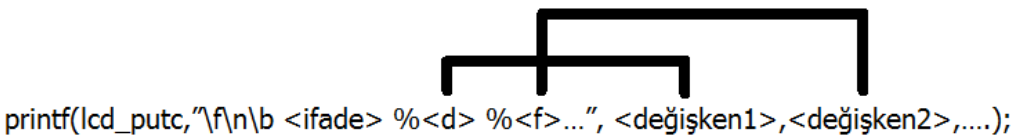
5. lcd_send_byte(byte adres, byte bilgi)

LCD ye istenen bir komutu gönderir.

6. lcd_read_byte ( )

LCD den 1 bayt veri okur

lcd_putc ( ) komutu yerine printf(lcd_putc,"….") komutu kullanılabilir. printf komutu C dilinde olan bir komuttur. printf komutu ile ekranda gösterilecek değişkenlerin formatı ayarlanabilir. Komutun genel kullanım formatı aşağıdaki gibidir.



Şekil 1.59: prinf komutunun genel kullanımı

Printf komutu içerisine ilk başta lcd_putc komutu yazılır sonra virgül konur. Çift tırnak açılarak gerekli bilgi ve sabitler yazılır. lcd_putc komutunda olduğu gibi ters eğik çizgi sabitleri (\f, \b,\n) kullanılabilir. **%** işaretinden sonra gelen kod değişkenin ekranda hangi formatta gösterileceğini belirtir. Yukarıdaki örnekte **% d** ile belirtilen yere değişken1' in içeriği tam sayı tipinde yazılır. **%f** ile belirtilen yere ise değişken2 ' nin içeriği ondalıklı tipte yazılır. Aşağıdaki örnekte volt isimli bir değişkenin içeriği lcd ye yazdırılmaktadır.

Örnek: printf(lcd_putc,"\f \nGerilim=%d V", volt);

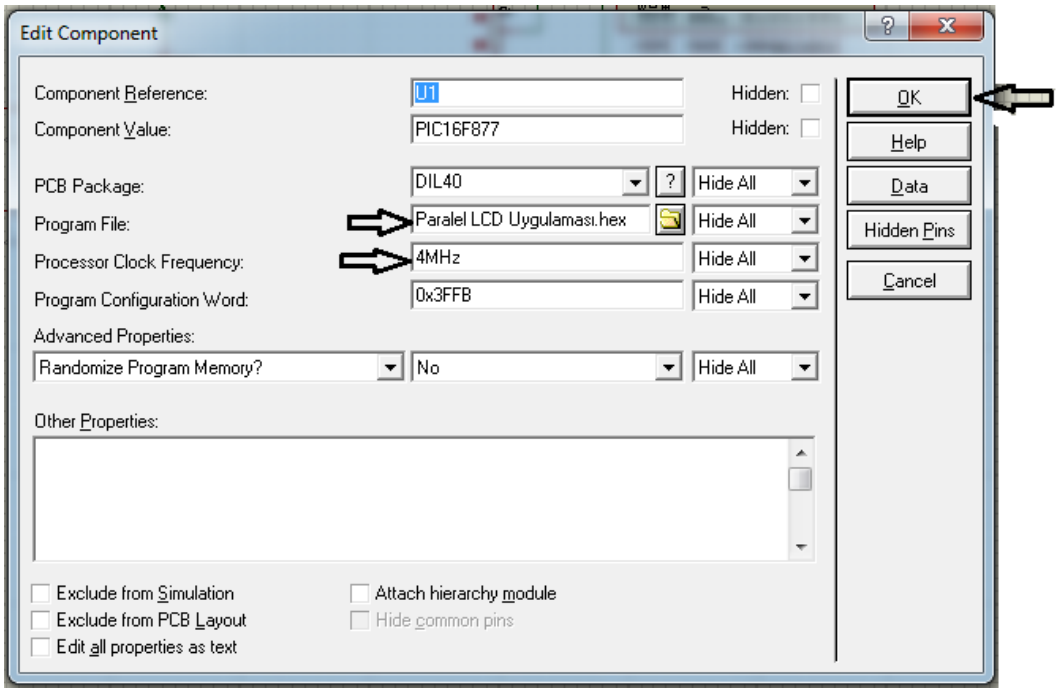
Örnekte ilk başta **\f** komutu ile LCD silinmektedir. Ardından **\n** komutu ile satır atlanarak 2. satıra geçilmiştir. LCD ye **Gerilim=** yazısı yazdırılmış, ardından **%d** komutunun olduğu yere **volt** değişkenin içeriği tam sayı formatında yazdırılmıştır. Sonra 1 karakter boşluk bırakılarak **V** harfi LCD ye yazdırılmıştır. Örneğin **volt** değişkeninin içeriğinin 20 olduğunu varsayarsak LCD de aşağıdaki görüntü oluşur.

Gerilim=20 V

1.3.1.5. Hex Kodunun Proteus ISIS ile Simülasyonu

Mikrodenetleyici ve çevresel bağlantıları yapıldıktan sonra PIC C programı tarafından üretilen hex kodunun yolunun gösterilmesi ve denetleyiciye bağlanan osilatörün frekansının belirtilmesi gerekmektedir. Denetleyici sembolünün üzerine sağ tıklanır, açılır menüden Edit Properties seçilir, Şekil 1.60 daki pencereden ilgili yerler doldurulur. Isis programı osilator frekans değeri olarak bu alana girilen değeri esas alır. Isis programının düzgün çalışması için denetleyiciye harici bir osilatör bağlanmamış olsa bile bu alana osilatörün frekansının doğru bir şekilde yazılması yeterlidir. Son olarak OK düğmesine tıklanarak yapılan değişiklikler kaydedilir.

Simülasyonu çalıştırmak için Şekil 1.61 görülen Isis programının sol alt köşesinde bulunan Play düğmesine, simülasyonu durdurmak için ise Stop düğmesine basılması gerekir.



Şekil 1.60: ISIS Programında Denetleyici Ayarları

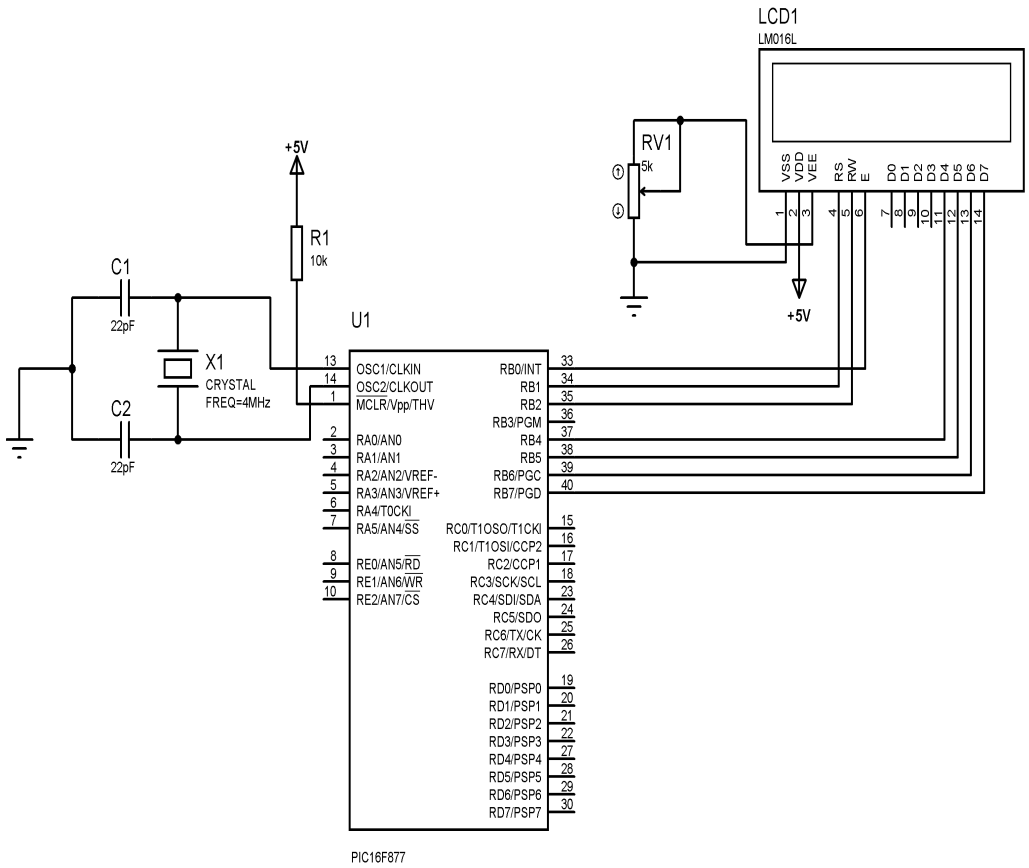
Şekil 1.61: ISIS Programı çalıştırma çubuğu

1.3.2. Paralel LCD Uygulamaları

1.3.2.1.Çeşitli Paralel LCD Komutları ve Uygulaması

ÖRNEK 1.1: Bu uygulamada CCS C programının hazır lcd.c dosyası kullanılarak uygulama yapılacaktır. 2 satır, 16 sütun (2x16) LCD kullanılacaktır. Şekilde gösterildiği gibi LCD B portuna bağlı olduğu standart bağlantı kullanılmıştır. Programda LCD komutlarının birçoğu kullanılacaktır.

Bağlantı Şeması:



Şekil 1.62: Örnek 1.1 İçin Bağlantı Şeması

Not: Çift sağa yatık çizgi // işaretinden sonra italik formda gösterilen yazılar program tarafından dikkate alınmaz açıklama yapmak amacı ile kullanılmıştır. Programın düzgün çalışması için **kalın** harfle gösterilenlerin yazılması zorunludur. Bu dosyadan kopyala yapıştır yapılırsa program bazı karakterleri tanımayıp hata verebilir. Programda İ, Ü gibi Türkçe harfleri kesinlikle kullanmayınız. LCD CGROM da Türkçe karakter tanımlı olmadığından anlamsız karakterler görünecektir.

Örnek1.1'in Program Kodu:

```
/*******************************************************
PIC16F877 ile
Çeşitli Paralel LCD Komutları ve Uygulaması
*******************************************************/

#include <16f877.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.

// Denetleyici konfigürasyon ayarları
#fuses  XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
 NODEBUG,NOCPD

#use delay (clock=4000000) // osilatör frekansı belirtiliyor.

#define use_portb_lcd TRUE   // LCD B portuna bağlı

#include <lcd.c>   // lcd.c dosyası tanıtılıyor

int a;      // Tamsayı tipinde değişken tanımlanıyor
float g;    // Ondalıklı tipte değişken tanımlanıyor
char k;     //  Karakter tipinde değişken tanımlanıyor

//********** ANA PROGRAM FONKSİYONU*******

void main ( )
{
setup_psp(PSP_DISABLED);       // PSP birimi devre dışı
setup_spi(SPI_SS_DISABLED);    // SPI birimi devre dışı
setup_timer_1(T1_DISABLED);    // T1 zamanlayıcısı devre dışı
setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);   // ANALOG giriş yok
setup_adc(ADC_OFF);            // ADC birimi devre dışı
setup_CCP1(CCP_OFF);           // CCP1 birimi devre dışı
setup_CCP2(CCP_OFF);           // CCP2 birimi devre dışı

lcd_init();   // LCD hazırlanıyor
a=5;   // x değişkenine değer veriliyor
g=2.8; // y değişkenine değer veriliyor
while(1) // Sonsuz döngü
{
lcd_send_byte(0,0x0F);     // LCD'ye "imleç gösterilir, imleç yanıp söner komutu" gönderiliyor.
lcd_putc("\fLULEBURGAZ"); // LCD'ye "LULEBURGAZ" verisi gönderiliyor.
delay_ms(1000); // 1 sn gecikme
lcd_putc("\n      EML"); //2. satırda 4 adet boşluk verildikten sonra "EML" stringi yazdırılıyor.
```

```
        delay_ms(1000); // 1 sn gecikme
        lcd_send_byte(0,0x0C); // LCD'ye "imleç gösterilmez, imleç yanıp sönmez" komutu gönderiliyor.
        lcd_gotoxy(1,2);   // Kursör 1.sütun, 2.satıra konumlandırılıyor.
        printf(lcd_putc,"\fAKIM=%d A",a); // LCD silinip AKIM=5 A ifadesi yazdırılıyor.
        delay_ms(1000); // 1 sn gecikme
        printf(lcd_putc,"\nGERILIM=%f V",g);    // 2. satıra GERILIM=2.80 V ifadesi yazdırılıyor.
        delay_ms(1000); // 1 sn gecikme
        printf(lcd_putc,"\fFREKANS"); // LCD silinip "FREKANS" stringi yazdırılıyor.
        k=lcd_getc(6,1);   // LCD'nin 6.Sütun ve 1.Satırdaki karakter alınıp k değişkenine aktarılıyor.
        printf(lcd_putc,"\n6.HARF=%c",k); // k değişkeni LCD'ye aktarılıyor(Şekil )
        delay_ms(1000); // 1 sn gecikme
        lcd_send_byte(0,0x01); // LCD silme komutu gönderiliyor.      lcd_send_byte(0,0x0d); // LCD'ye "imleç gösterilmez, imleç yanıp söner" komutu gönderiliyor.
        lcd_gotoxy(10,1); // Kursör 10.sütun, 1.satıra konumlandırılıyor.
        lcd_send_byte(0,0x04); // Kursörü sola kaydıran komut gönderiliyor.
        lcd_putc("TERS_YAZI");  // "TERS_YAZI" stringi sağdan sola doğru yazdırılıyor.
        delay_ms(1000);  // 1 sn gecikme
        }
        }/////Program sonu
```

Şekil 1.63: Örnek 1.1 İçin LCD'ye gönderilen veriler

Örnek 1.1 Açıklama: Her komutun ardından ekran çıktılarının 1 saniye süreyle görünmesi için delay_ms(1000); komutu gönderilmiştir. Parantez içindeki değer ms cinsinden gecikme süresini ayarlar. Bu değeri değiştirerek gecikme süresi ayarlanabilir. Programda çalışması istenmeyen komutların başına çift slaş işareti **//** konulabilir**,** veya çalışması istenmeyen program parçacıkları slaş-yıldız /* ve yıldız-slaş */ işaretleri arasına alınabilir. Böylece incelemeye gerek duyulmayan, çalışması istenmeyen kısımlar üzerinde zaman kaybedilmemiş olur.

#define use_portb_lcd TRUE ifadesi ile denetleyicinin B portunun LCD ye bağlı olduğu bildirilmiştir. Eğer LCD D portuna bağlanmış olsaydı herhangi bir komut kullanımına gerek kalmayacaktı.

#include <lcd.c> komutu ile lcd.c sürücü dosyası programa dahil edilmiştir.

int a komutu ie tam sayı tipi**, float g** komutu ile ondalıklı sayı, **char k** komutu ile karakter tipi değişkenler tanımlanmıştır.

Ana program kısmında **lcd_init()** komut ile LCD kullanıma hazırlanmıştır. Diğer LCD komutlarından önce bu komutun mutlaka kullanılması gerekir.

a=5 ve g=2.8 komutları ile değişkenlere değer atanmıştır.

while(1) ile sonsuz döngü oluşturulmuştur.

lcd_send_byte(0,0x0F) komutu ile LCD'ye "imleç gösterilir, imleç yanıp söner komutu" gönderilmiştir. Şekil de ki diğer komutlar da gönderilebilir, ancak display kapalı özellikli komutlar gönderilirse ekranda bir şey görünmemesine rağmen veriler DDRAM hafızasında saklı tutulur.

lcd_putc("\fLULEBURGAZ") komutu ile ilk önce **\f** sabiti ile ekran temizlenmiş boşluk verilmeden 1. Satır 1.sütuna **LULEBURGAZ** stringi yazdırılmıştır.

lcd_putc("\n EML") komutu ile ilk önce **\n** sabiti ile satır atlanarak 2. satıra geçilmiş, ardından 4 adet boşluk bırakılarak **EML** stringi yazdırılmıştır.

lcd_send_byte(0,0x0C) komutu ile LCD'ye "imleç gösterilmez, imleç yanıp sönmez" komutu gönderilmiştir.

lcd_gotoxy(1,2) komutu ile imleç 1.sütun, 2.satıra konumlandırılmıştır. **printf(lcd_putc,"\fAKIM=%d A",a)** komutu ile ilk önce **\f** sabiti ile ekran temizlenmiş, boşluk verilmeden **AKIM=** stringi yazdırılmış**, %d** ifadesinin olduğu yere **a** değişkenin değeri tam sayı tipinde yazdırılmış ve 1 boşluk bırakılarak **A** karakteri gösterilmiştir.

printf(lcd_putc,"\nGERILIM=%f V",g); komutu ile ilk önce **\n** sabiti ile satır atlanarak 2. satıra geçilmiş, boşluk verilemeden **GERILIM=** stringi yazdırılmış**,** ardından **%f** sabitinin olduğu yere **g** değişkenin içeriği ondalıklı olarak yazdırılmış, bir boşluk verilerek **V** karakteri gösterilmiştir.

printf(lcd_putc,"\fFREKANS"); ekran temizlenerek 1.satır 1. sütuna **FREKANS** stringi yazdırılmıştır.

k=lcd_getc(6,1); ile LCD nin ekranında o an için 1. Satır 6. Sütunda görünen karakter alınarak **k** değişkenine atanmıştır.

printf(lcd_putc,"\n6.HARF=%c",k); satır atlanarak 2. satır 1. sütuna **6.HARF=** stringi yazdırılmış **%c** sabitinin olduğu yere karakter tipindeki **k** değişkenin içeriği yazdırılmıştır**.**

lcd_send_byte(0,0x01); komutu ile ekran temizlenmiştir. Silme işlemi lcd_putc("\f") komutu ile de yapılabilir.

lcd_send_byte(0,0x0d); komutu ile LCD'ye "imleç gösterilmez, imleç yanıp söner" komutu gönderiliyor.

lcd_gotoxy(10,1); komutu ile kursör 1. satır 10. sütuna konumlandırılıyor.
lcd_send_byte(0,0x04); 0x04 ile kursörü her karakter girişinden sonra bir adım sola kaydıran komut gönderiliyor.

lcd_putc("TERS_YAZI"); komutu ile "**TERS_YAZI**" stringi LCD ye gönderiliyor. **0x04** komutandan dolayı ilk harf 10. Karakterin bulunduğu yere, 2. harf 9. karakterin bulunduğu yere, diğer harflerde yine sağdan sola doğru olacak şekilde konumlandırılıyor. Ekranda **IZAY_SRET** şeklinde çıktı oluşuyor.

ÖRNEK 1.2: Kayan Yazı Uygulaması

Şekil 1.62 deki bağlantı şeması kullanılmıştır. Bu uygulamada displayin sola (**0x18**) ve sağa (**0x1c**) kaydırma komutlarının uygulaması yapılacaktır.

Örnek 1.2'nin Program Kodu:

```
/*******************************************************
PIC16F877 ile
Kayan Yazı Uygulaması
*******************************************************/

#include <16f877.h>    // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.

// Denetleyici konfigürasyon ayarları
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
NODEBUG,NOCPD

#use delay (clock=4000000) // osilatör frekansı belirtiliyor.

#define use_portb_lcd TRUE   // LCD B portuna bağlı

#include <lcd.c>   // lcd.c dosyası tanıtılıyor

void sola_kaydir() // DDRAM içeriğini sola kaydırma fonksiyonu tanımlanıyor
{
lcd_send_byte(0,0x18); // DDRAM içeriğini sola kaydırma komutu veriliyor.
}

void saga_kaydir() // DDRAM içeriğini sağa kaydırma fonksiyonu tanımlanıyor
{
lcd_send_byte(0, 0x1c); // DDRAM içeriğini sağa kaydırma komutu veriliyor.
}
//********** ANA PROGRAM FONKSİYONU*******
void main()
```

```
{
int i, a; // i ve a adında tam sayı tipi değişkenler tanımlanıyor.
lcd_init(); //lcd baslat
lcd_putc("\f");  //lcd ekranı temizle
lcd_putc("   LULEBURGAZ EML ELEKTRIK-ELEKTRONIK   ");  // çift tırnak arasına gösterilecek string yazılıyor.
for(;;) // sonsuz döngü oluşturuluyor
{
for(a = 0; a<30; a++)  //30 karakter için sağa kaydırma
{
saga_kaydir();  // DDRAM içeriğini sağa kaydır fonksiyonu çalıştırılıyor
delay_ms(200);   // sağa kayma hızı
}

for(i = 0; i <30; i++)  // 30 karakter için sola kaydırma

sola_kaydir();  // DDRAM içeriğini sola kaydır fonksiyonu çalıştırılıyor
delay_ms(200);  //sola kayma hızı
}
}
}
```

Örnek 1.2 Açıklama:

void sola_kaydir() ifadesi ile **sola_kaydir** isminde bir fonkisyon tanımlanıyor**.** Süslü parantez içerisine **lcd_send_byte(0,0x18)** DDRAM içeriğini sola kaydırma komutu yazılıyor.

void saga_kaydir() ifadesi ile **saga_kaydir** isminde bir fonkisyon tanımlanıyor**.** Süslü parantez içerisine **lcd_send_byte(0,0x1C)** DDRAM içeriğini sağa kaydırma komutu yazılıyor.

int i, a; komutu ile ana programda döngü sayaçlarında kullanılacak değişkenler tanımlanıyor.

lcd_putc(" LULEBURGAZ EML ELEKTRIK-ELEKTRONIK "); komutu ile LCD gösterilmesi istenen string yazılıyor. 40 adet karakterden fazlası DDRAM e kaydedilmez ve ekranda görüntülenmez. Bu yüzden bir satıra ait karakter sayısının 40 ‘ı aşmaması gerekir.

for(;;) komutu ile başlangıç değeri, şart ve adım sayısı belirtilmediğinden sonsuz döngü oluşturulmuştur.

for(a = 0; a<30; a++); burada **a=0** ile döngünün başlangıç değerinin 0 olduğu**, a<30** ile döngü şartı tanımlanmış ve **a** değişkeni **30** dan küçük olduğu sürece devam edecektir**, a++** ifadesi ile sayaç değişkeni tanımlanmış ve a değişkeninin içeriğinin her döngüde 1 kez attırılacağı belirtilmiştir. Döngü çalışma süresine a değeri 0 ila 29 arasında 30 farklı değer alacaktır. a=30 olduğunda döngü şartı sağlanmadığından döngüden çıkılacaktır. Döngü şartı olarak a<30 ifadesinde 30 sayısı yerine farklı değerler yazılabilir. Örneğin a<1 yazılırsa DDRAM içeriği bir kez sağa kayacaktır, bu yüzden tanımlanan stringin bir kısmı görüntülenemeyecektir.

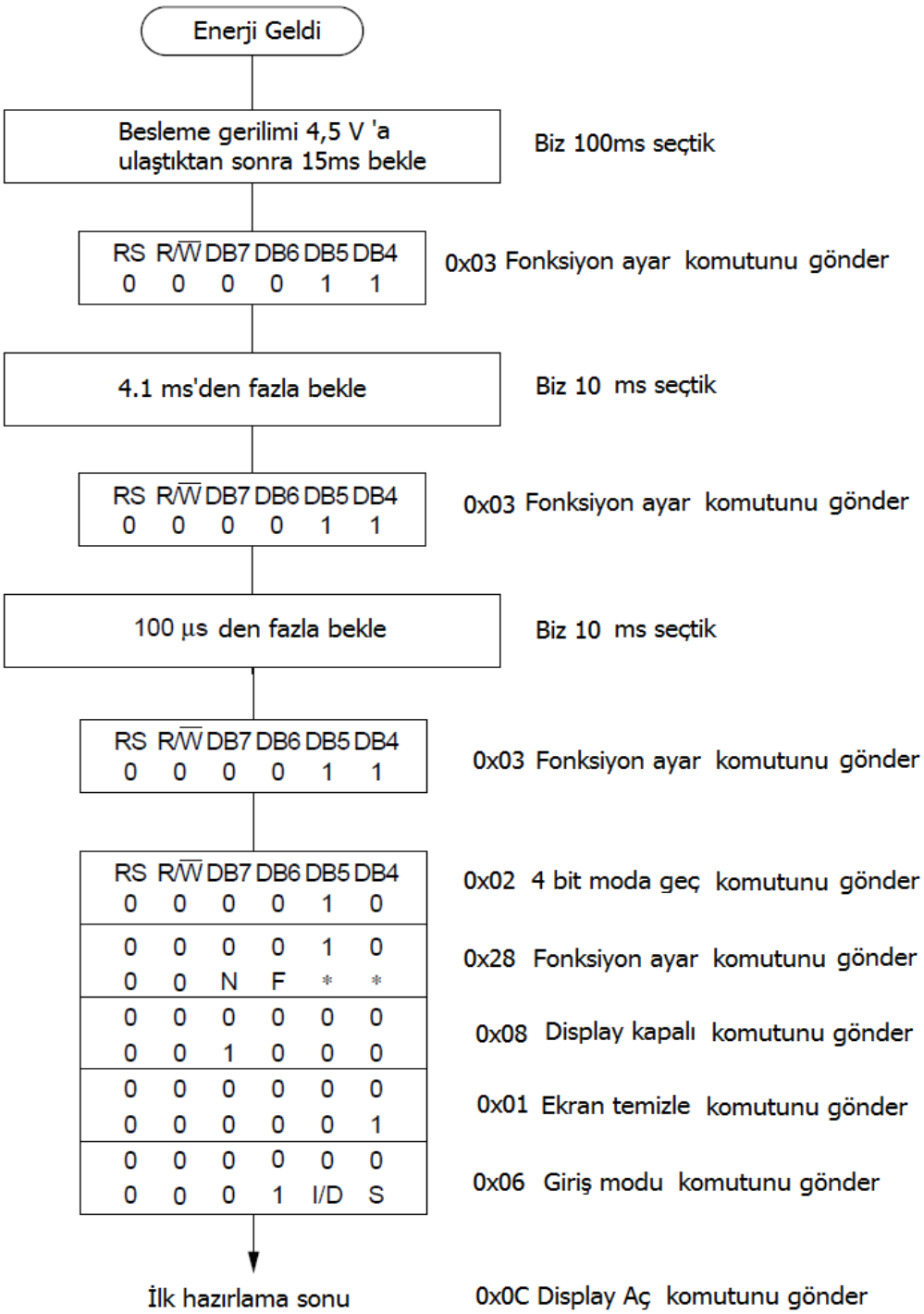
saga_kaydir(); komutu ile DDRAM içeriğini sağa kaydır fonksiyonu çalıştırılıp LCD ye 0x1C sağa kaydır komutu gönderiliyor. Burada sağa kaydırılan DDRAM içeriğidir. Bu yüzden yazılar sola kayıyormuş gibi görünecektir. Şekil bakarak DDRAM adres yapısını tekrar hatırlayınız.

delay_ms(200); komutu ile kayma hızı ayarlanmaktadır. Parantez içerisine daha küçük değerler girilirse kayma hızı artacak, daha büyük değerler girilirse kayma hızı yavaşlayacaktır.

Programın geri kalan kısmında sola kaydırma işlemi için for döngüsü oluşturulmuş ve benzer adımlar tekrar edilmiştir.

1.3.2.2. Farklı bir LCD Sürücü Dosyası Tasarımı ve Uygulaması

HD44780 LCD içinde dahili reset (internal reset circuit) devresi vardır. LCD'ye enerji uygulandığı zaman dahili reset devresi çalışırak gerekli komutları ve işlemleri otomatik olarak gerçekleştir. Dahili reset devresinin görevini gerçekleştirebilmesi için besleme kaynağının yükselme zamanın (0' dan 4,5V gelme süresi) 1 ila 10 ms arasında olması ve kapanma süresinin 1ms ve altında olması gerekir. Eğer besleme kaynağı bu şartları sağlamıyorsa dahili reset devresi düzgün çalışmaz. Bu durumda reset işleminin kullanıcı tarafından programla yapılması gerekir. Şekil 1.66 da HD44780 LCD'nin 4 bit çalışma modu için üreticisi tarafından önerilen akış diyagramı önerilmektedir. Akış diyagramından görüleceği üzere enerji geldikten sonra gerilimin 4,5 V 'a yükselene kadar 15ms beklenmesi öneriliyor. Bazı kaynaklar bekleme süresini 40ms olarak önermektedir. Bekleme süresini 100 ms olarak seçtik. LCD ilk önce 8 bit modunda çalışmaya başlatılır. 8 bit çalışma modu fonksiyon ayar komutu olarak DB5 ve DB4 pinlerine 3 defa lojik 1 sinyali gönderilmesi ve bu sinyaller arasında 4,1ms ve ardından 100 µs beklenmesi önerilmektedir. Burada bekleme süresini 10ms olarak seçtik. Daha sonra DB5 pinine lojik 1 ve DB4 pinine Lojik 0 komutu gönderiliyor. Bu komut LCD nin 4 bit modta çalışmaya başlamasını sağlar. Sonraki adımda fonksiyon ayar komutu olarak 0x28 komutu (4 bit iletişim, LCD 2 satır, 5x7 dot matris) gönderiliyor. Sonraki adımda 0x08 komutu (Display kapalı, imleç gösterilmez, imleç yanıp sönmez) komutu gönderiliyor. Bir sonraki adımda 0x01 komutu (ekran silme ve imleci 1. Satır, 1. Sütuna yerleştir) komutu gönderiliyor. Daha sonra giriş modu komutu olarak 0x06 h (Gösterilen her bir karakterden sonra imleci bir sağa kaydır) gönderiliyor. Bu aşamada display kullanıma hazır hale gelmiştir. En son olarak 0x 0C komutunu (Display açık, imleç gösterilmez, imleç yanıp sönmez) gönderiyoruz.



Şekil 1.64: LCD nin ilk açılış ayarları

LCD Kontrolünde Dikkat Edilecek Hususlar

Mikrodenetleyici ile LCD kontrolünde birtakım unsurlara dikkat edilmelidir. RS, R/W ve E uçlarına uygun lojik değerler uygulanmalıdır.

LCD'ye veri veya komut gönderilecekse R/W ucu lojik-0 olmalı, LCD'den bir veri okunacaksa R/W ucu lojik-1 yapılmalıdır.

Eğer komut gönderilecek ise önce RS ucu lojik-0 yapılmalı sonra komut gönderilmelidir. Veri gönderilecek ise RS ucu lojik-1 yapılmalıdır.

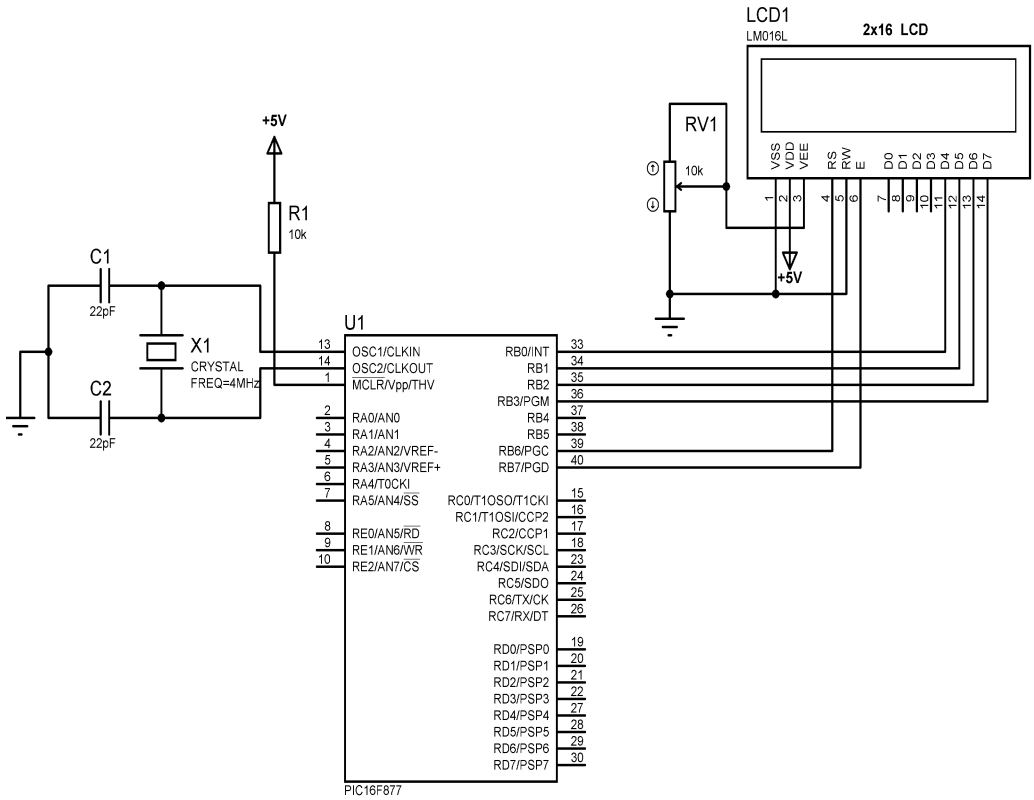
Bir komutun veya verinin LCD içerisinde işleme konması için E ucuna düşen kenarlı bir izin darbesi uygulanmalıdır.

Komut uygulama zamanlarına dikkat edilmelidir. Bir komut uygulandıktan sonra daha önce belirtilen komut icra süreleri kadar bekleme yapmak gereklidir. En kesin çözüm LCD'nin meşgul olup olmadığını kontrol etmektir. BF meşgul bayrağı daha önce anlatıldığı gibi kontrol edilerek bu gerçekleşir. Böylece en kısa zaman gecikmesi elde edilir. Program sadeliği ve kolay anlaşılması bakımından delay komutu tercih edilmiştir.

Displayi sürmek için **bizim_LCD. c** isimli sürücü dosyası tasarlayacağız. Denetleyici ve LCD arasında tercih edilen bağlantı şekil 1.65 ve şekil 1.66 de görülmektedir. LCD ye sadece yazma yapılacağı için R/W pinini lojik 0 (şaseye bağladık) yaptık.

Mikrodenetleyici Pin Adı	LCD Pin Ad
B0	D4
B1	D5
B2	D6
B3	D7
B6	R/S
B7	E
Şaseye bağlantı	R/W

Şekil 1.65: Denetleyici LCD pin bağlantıları



Şekil 1.66: Örnek 1.3 için bağlantı şeması

Örnek 1.3: Farklı bir LCD Sürücü Dosyası Tasarımı ve Uygulaması (bizim_LCD.c sürücü programı)

```
//*******Farklı bir LCD Sürücü Dosyası Tasarımı ve Uygulaması
//********************* bizim_LCD.c ***************************

#define rs  pin_b6 // LCD'nin RS ucu RB6 pinine bağlı
#define e   pin_b7 // LCD'nin E ucu RB7 pinine bağlı

//******LCD'ye Komut Gönderme Fonksiyonu **********

void lcd_komut(byte komut)
{
output_b(komut>>4);   // Komutun yüksek değerli 4 bitini gönder
output_low(rs);       // LCD komut almak için ayarlandı
delay_cycles(1);      // 1 komut saykılı bekle
output_high(e);       // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);      // 1 komut saykılı bekle
output_low(e);        // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);          // 5 msn gecikme veriliyor
output_b(komut&0x0F); // Komutun düşük değerli 4 bitini gönder
```

```
output_low(rs);      // LCD veri almak için ayarlandı
delay_cycles(1);     // 1 komut saykılı bekle
output_high(e);      // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);         // 5 msn gecikme veriliyor
}
//******* LCD'ye Veri Gönderme Fonksiyonu **********
void lcd_veri(byte veri)
{
output_b(veri>>4);   // Verinin yüksek değerli 4 bitini gönder
output_high(rs);     // LCD veri almak için ayarlandı
delay_cycles(1);     // 1 komut saykılı bekle
output_high(e);      // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(1);         // 1 msn gecikme veriliyor
output_b(veri&0x0F); // Verinin düşük değerli 4 bitini gönder
output_high(rs);     // LCD veri almak için ayarlandı
delay_cycles(1);     // 1 komut saykılı bekle
output_high(e);      // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(1);         // 1 msn gecikme veriliyor
}

//******* LCD'de İmlec Konumlandırma Fonksiyonu ********
void imlec(byte satir, byte sutun)
{
if (satir==1)     // Eğer 1. satır seçili ise
lcd_komut(0x80|(sutun-1));

if (satir==2)     // Eğer 2. satır seçili ise
lcd_komut(0xC0|(sutun-1));
}

//********* LCD Başlangıç Ayarları Fonksiyonu ******
void lcd_hazirla()
{
int i=0;
output_low(rs); // RS ucu lojik-0
output_low(e);  // E ucu lojik-0
delay_ms(100);   // LCD enerjlendiğinde LCD'nin hazır olması için beklenen süre
for(i=0;i<3;i++)    // LCD'ye 3 kez 0x03 fonksiyon komutu gönderiliyor
{
lcd_komut(0x03);
delay_ms(10);     // 10 msn gecikme veriliyor
```

```
}
lcd_komut(0x02); // LCD'ye 4 bit iletişim komutu gönderiliyor
lcd_komut(0x28); // 4 bit iletişim, 2 satır, 5x8 dot matris seçildi
lcd_komut(0x08); // Display Kapalı, imleç gösterilmez ve yanıp sönmez
lcd_komut(0x01); // Display sil. İmleç 1.satır 1.sütunda
lcd_komut(0x06); // Her gönderilen karakterden sonra imleç bir sağa gitsin
lcd_komut(0x0C); // Display açık,imleç gösterilmez ve yanıp sönmez.
}
```

Açıklama:

#define rs pin_b6 ve **#define e pin_b7** komutları ile B6 pinine **rs,** B7 pinine **e** isimleri atanmıştır.

void lcd_komut(byte komut) ifadesi ile LCD ye 1 baytlık (8 bit) komut gönderme fonksiyonu tanımlanmıştır. LCD ile 4 bitlik haberleşme yapılacağından 8 bitlik komut bilgisinin ilk önce yüksek 4 biti (MSB), daha sonra ise düşük 4 biti (LSB) gönderilecektir.

output_b(komut>>4); komutu ile komut bitleri 4 kez sağa kaydırılmıştır. Böylece MSB bitleri LSB bitlerinin üzerine gelmiştir. Örnek olarak 8 bitlik komutumuz 0x28 h ise 8 bitin durumu 00101000 olacaktır**. >> 4** komutu ile 4 kez sağa kaydırma işlemi sonrasında komut değeri 00000010 şeklini almıştır ve bu değer B portuna gönderilecektir. Hatırlanacağı üzere >> komutu ile kaydırma işlemi sonrası boşalan yerler 0 ile beslenmektedir.

output_low(rs); komutu ile LCD komut almak için ayarlandı. Komutun uygulanması için E ucundan (**E**nable) düşen kenar sinyali uygulanması gerekmektedir. Bunun için **e** ucu ilk önce **output_high(e);** komutu ile lojik 1 yapılıp daha sonra **output_low(e);** komutu ile lojik 0' a çekilmektedir. Şekil den görüleceği üzere her komutun bir uygulama süresi vardır. Buradaki değerler 8 bit çalışma modu için verilmiştir. Uygulama süresi en uzun olan komut **ekran sil** komutu olup uygulama zamanı için önerilen süreler 8bit mod için 82µs-1.64ms arasında, 4 bit mod için 120µs- 4.9ms arasında değişmektedir. En olumsuz şartları göz önüne alarak uygulama süresini 5ms seçtik ve **delay_ms(5);** komutu ile bu gecikmeyi sağladık. **output_b(komut&0x0F);** komutu ile komut bilgisinin düşük değerlikli 4 biti (LSB) B portuna gönderilmiştir. Örneğin 8 bitlik komutumuz 0x28 h ise 8 bitin durumu 00101000 şeklindedir. **komut&0x0F** işleminin uygulanmasından sonra yüksek değerlikli 4 bit (MSB) lojik **VE** (&) operatörü ve 0 ile maskelenip lojik 0 olacak, alt 4 bit olduğu gibi kalacaktır. Komutumuz 00001000 şeklini alacaktır. Daha önce bahsedildiği gibi her işlemin gerçekleşmesi için **e** ucuna düşen kenar tetiklemesi uygulamak için sırası ile **output_high(e); ve output_low(e);** komutları çalıştırılmıştır.

void lcd_veri(byte veri) komutu ile LCD ye bir baytlık veri gönderme fonksiyonu oluşturulmuştur. output_b(veri>>4); komutu ile veri değerinin yüksek 4 biti LCD' ye gönderilmiştir. output_high(rs); komutu ile LCD veri almak için ayarlanmıştır. Daha önce bahsedildiği gibi her işlemin gerçekleşmesi için e ucuna düşen kenar tetiklemesi uygulamak için sırası ile output_high(e); ve output_low(e); komutları çalıştırılmıştır. LCD ye veri yazma süresi için 8bit modta 40µs, 4 bit modta ise 120µs süresi bırakılması önerilmektedir.

Yine en olumsuz şartları göz önüne alarak veri yazma için bekleme süresini 1ms olarak seçtik ve delay_ms(1); komutu ile yeterli gecikmeyi sağladık.

output_b(veri&0x0F) komutu ile veri değerinin düşük 4 biti LCD' ye gönderilmiştir. Yüksek 4 bitin gönderilmesi için yapılanlar düşük 4 bitin gönderilmesi için tekrarlanmıştır.

void imlec(byte satir, byte sutun) komutu ile imleci konumlandırma fonksiyonu tanımlanmıştır. Daha önceden de bahsedildiği gibi imleci konumlandırmak için DDRAM adres kurma komutunun kullanılması gerekir. İmleci 1. satır, 1. sütuna konumlandırmak için DDRAM'e 0x80h verisi gönderilir. İmleci 2. satır, 1. sütuna konumlandırmak için DDRAM'e 0xC0 h verisi gönderilir. Örnek olarak imleci 1. satır ve 8. sütununa konumlandırmak için 1. Satırın ilk sütununun adres kurma komutuna (80H) sütun sayısının 1 eksiği (8-1) ilave edilir. Bu durumda LCD'ye 0x87 H komutunun gönderilmesi gerekir. Diğer bir örnek olarak imleci 2. satır ve 5. sütunu konumlandırmak için 2. Satırın ilk sütununun adresine (C0h) sütun sayısının 1 eksiği (5-1) ilave edilir. Bu durumda 84H komutunun gönderilmesi gerekir. **if (satir==1)** ifadesi ile satır değişkeni 1 ise veya 1. Satır seçilmiş ise **lcd_komut(0x80|(sutun-1));** komutunun işlem görmesi sağlanmıştır. Burada sütun sayısının 1 eksiği ile 1.satır 1. sütunun adres değerine (0x80) mantıksal **VEYA (OR)** işlemi uygulanmıştır. Bilindiği üzere mantıksal VEYA (OR) işlemi toplamaya denk bir işlemdir. Elde edilen sonuç komut gönderme fonksiyonu **lcd_komut** () ile LCD'ye gönderilmiştir. **if (satir==2)** ifadesi ile 2. satır seçili ise **lcd_komut(0xC0|(sutun-1));** komutunun işlem görmesi sağlanmıştır. Burada sütun sayısının 1 eksiği ile 2.satır 1. sütunun adres değerine (0xC0) mantıksal **VEYA (OR)** işlemi uygulanmıştır. Çıkan sonuç yine komut gönderme fonksiyonu **lcd_komut** () ile LCD'ye gönderilmiştir. Konuyu daha iyi anlamak için Paralel LCD Bellek Yapıları ve DDRAM Adres Kur başlıkları altındaki konuları tekrar ediniz.

void lcd_hazirla() fonksiyonu ile LCD'nin başlangıç ayarları yapılmıştır. Şekilde görülen akış diyagramı bu fonksiyon ile uygulamaya geçirilmiştir. **int i=0;** ile i isminde tam sayı tipi bir değişken tanımlanmış ve ilk değer olarak 0 atanmıştır. **output_low(rs);** komutu ile rs ucu lojik 0 yapılmıştır. **output_low(e);** komutu ile e ucu lojik 0 yapılmıştır. **for(i=0;i<3;i++)** komutu ile ilk değeri 0, döngü şartı i değerinin 3'ten küçük olması, döngü adımı ise 1 olan döngü oluşturulmuştur. Akış diyagramından görüldüğü gibi LCD'nin ilk açılışta 8 bit modunda açılması gerekir, bunun için fonksiyon ayar komutu olarak DB5 ve DB4 pinlerinin her ikisi lojik 1 yapılır, bu komutun ikili sayı sistemindeki karşılığı 110000, hex karşılığı 30 h tır. Komut gönderme foksiyonumuz 4 bit komut göndermeye göre tasarlanmıştır. 30 h komutunu **lcd_komut(0x30);** şeklinde göndermiş olsaydık ilk önce yüksek 4 bit gönderilecek yani 3h sayısı gönderilip DB5 ve DB4 pinleri lojik 1 olacak, akabinde düşük 4 bit gönderilecek yani 0h sayısı gönderilip DB5 ve DB4 pinleri lojik 0 yapılacak ve lojik 0 olarak kalacaktı. DB5 ve DB4 pinlerini lojik 1 yapmak isterken tam tersi lojik 0 yapmış olduk. **lcd_komut**() fonksiyonun tasarımından dolayı 8 bit mod çalışmada üst 4 bitin önemsiz olduğunu, alt 4 bitin LCD' ye yazıldığını öğrenmiş olduk. Bu engeli aşmak için basit bir hileye başvuracağız; 3 h sayısını alt 4 bite yazarak lcd_komut(0x03); komutunu LCD' ye göndereceğiz. delay_ms(10); komutu ile komutun uygulanması için 10 ms gecikme sağlanmıştır. For döngüsü sayesinde arka arkaya 3 kez 0x03 komutu gönderilmiş olup her gönderim sonrası 10 ms gecikme yapılmıştır. Akış diyagramından görüleceği üzere 4 bit moda geçmek için DB5 pinin lojik 1, DB4 pininin Lojik 0 yapılması yani 20 h sayısının

LCD'ye gönderilmesi öneriliyor. 8 bit modta çalışmaya devam edildiği için 2h sayısını alt 4 bite yazıp komut gönderme fonksiyonunu **lcd_komut(0x02);** şeklinde düzenleyeceğiz. 0x02 komutunun uygulanmasından sonra 4 bit çalışma moduna geçilmiştir. **lcd_komut(0x28);** ile LCD ile denetleyicinin 4 bit iletişim yapacağı , LCD nin 2 satıra sahip olduğunu, her bir karakter için 5x8 dot matris fontuna sahip olduğunu LCD'ye bildirmiş olduk. Sonraki adımda **lcd_komut(0x08);** (Display kapalı, imleç gösterilmez, imleç yanıp sönmez) komutu gönderiliyor. Bir sonraki adımda **lcd_komut(0x01);** (ekranı temizle ve imleci 1. Satır, 1. Sütuna yerleştir) komutu gönderiliyor. Daha sonra giriş modu komutu olarak **lcd_komut(0x06);** (Gösterilen her bir karakterden sonra imleci bir sağa kaydır) komutu gönderiliyor. Bu aşamada display kullanıma hazır hale gelmiştir. En son olarak **lcd_komut(0x0C)** komutunu (Display açık, imleç gösterilmez, imleç yanıp sönmez) gönderiyoruz.

Oluşturduğumuz **bizim_LCD.c** dosyasını programlarımızda kullanabilmemiz için uzantısı ".C" olacak şekilde Drivers klasörü içine kaydetmemiz gerekmektedir. Bu klasörün yeri kurulum esnasında farklı bir yer seçilmediyse C:\Program Files\PICC\ Drivers şeklindedir. Bu dosyayı programlarımızda kullanabilmek için **#include < bizim_LCD.c** >satırını programımıza eklememiz gerekmektedir. Eğer dosyayı farklı bir klasörde tutmak istiyorsak klasörün yolunu belirtmemiz gerekir. Örneğin dosyamız C sürücüsü içerisinde Programlarim isimli bir klasör içindeyse **#include < C:\Programlarim\bizim_LCD.c >** satırını programlarımıza eklememiz gerekir.

Örnek 1.4: bizim_LCD.c Driver Dosyası ile LCD Uygulaması Program Kodu

```
/*********************************************************
bizim_LCD.c Dosyası ile LCD Uygulaması
*********************************************************/
#include <16f877.h>     // Denetleyicinin başlık dosyası tanıtılıyor.
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
NODEBUG, NOCPD  // Denetleyicinin konfigürasyon ayarları
#use delay (clock=4000000) // Kullanılacak osilatör frekansı.
#include <bizim_LCD.c> // bizim_LCD.c dosyası programa tanıtılıyor
/********************** ANA PROGRAM **********************/
void main ( )
{
setup_psp(PSP_DISABLED);        // PSP birimi devre dışı
setup_timer_1(T1_DISABLED);     // T1 zamanlayıcısı devre dışı
setup_timer_2(T2_DISABLED,0,1);  // T2 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);    // ANALOG giriş yok
setup_adc(ADC_OFF);             // ADC birimi devre dışı
setup_CCP1(CCP_OFF);            // CCP1 birimi devre dışı
setup_CCP2(CCP_OFF);            // CCP2 birimi devre dışı

lcd_hazirla();  // LCD nin ilk açılış ayarları yapılıyor

while(1)  // Sonsuz döngü
{
```

```
imlec(1,3);  // İmlec 1.satır 3.sütunda
printf(lcd_veri,"LULEBURGAZ");   // LCD'ye veri gönderiliyor
delay_ms(1500);
lcd_komut(0x0D);  // İmleç gösterilmez yanıp söner
imlec(2,7);  // İmlec 2.satır 7.sütunda
printf(lcd_veri,"E");  // LCD'ye veri gönderiliyor
delay_ms(1500);
printf(lcd_veri,"M");  // LCD'ye veri gönderiliyor
delay_ms(1500);
printf(lcd_veri,"L");  // LCD'ye veri gönderiliyor
delay_ms(1500);
lcd_komut(0x01);  // LCD temizleniyor
}
}
```

Şekil 1.67: Örnek 1.4 ekran çıktısı

Açıklama: #include <bizim_LCD.c> komutu ile bizim_LCD.c dosyası programa dahil edilmiştir. bizim_LCD.c dosyasında gecikme kullanıldığından #include <bizim_LCD.c> komutu #use delay (clock=4000000) komutundan sonra kullanılmalıdır. Tersi yapılırsa program hata verir. Ana programda lcd_hazirla( ); komutu ile LCD nin başlangıç ayarları yapılıyor. **imlec(1,3);** komutu ile imlec 1.satır 3.sütuna konumlandırılıyor. **printf(lcd_veri,"LULEBURGAZ");** komutu ile LCD'ye **LULEBURGAZ** stringi yazdırılıyor. **delay_ms(1500);** komutu ile 1500 ms gecikme sağlanmıştır. **lcd_komut(0x0D);** komutu ile imlecin durumu imleç gösterilmez, yanıp söner şeklinde ayarlanıyor. **imlec(2,7);** komutu ile imlec 2.satır 7.sütuna konumlandırılıyor. **printf(lcd_veri,"E");** komutu ile E karakteri 2.satır 7.sütuna yazlıyor. **printf(lcd_veri,"M");** komutu ile M karakteri 2.satır 8.sütuna yazlıyor.

printf(lcd_veri,"L"); komutu ile L karakteri 2.satır 9.sütuna yazlıyor.

M ve L karakterlerinin konumlandırılması otomatik olarak yapılıyor. Hatırlanacağı üzere DDRAM'e her karakter gönderildikten Adres Sayıcının (AC) içeriği 1 artmaktadır.

lcd_komut(0x01); komutu ile LCD temizleniyor ve program başa dönerek **while(1)** döngüsünün altından çalışmasına devam ediyor.

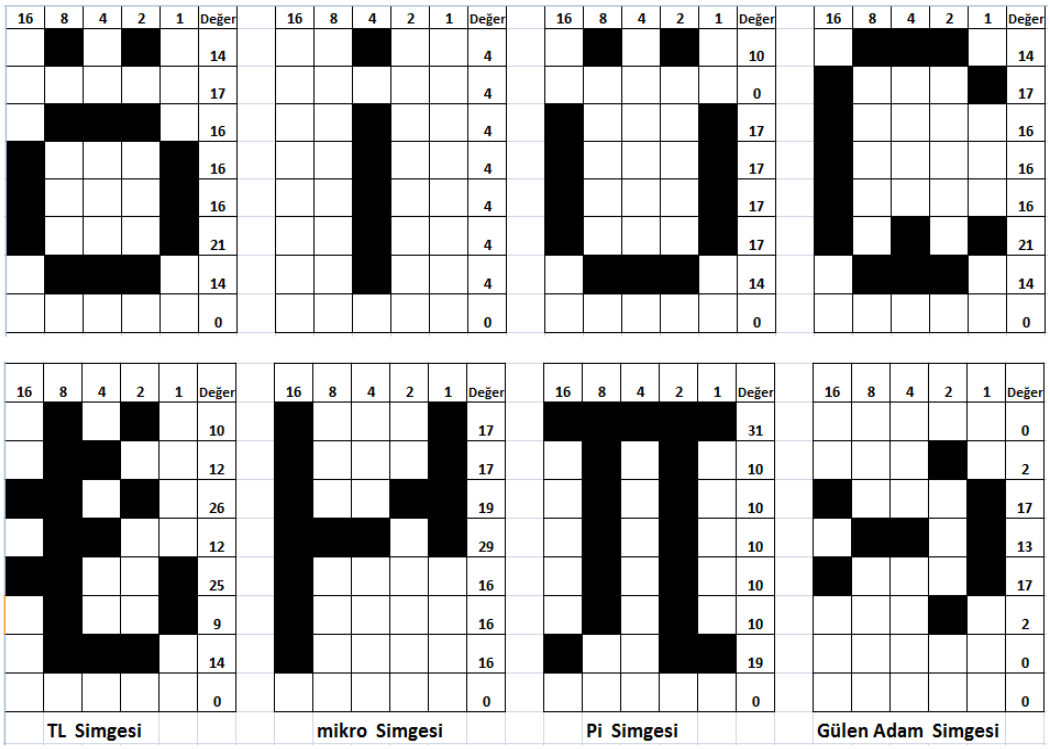
1.3.2.3. Türkçe ve Özel Karakterli LCD Dosyası Tasarımı ve Uygulaması

LCD de CGROM (Character Generator ROM) içerisinde silinemeyen, değiştirilemeyen standart karakter karakter grubu tanımlanmıştır. Kullanıcılar istedikleri karakterleri CGRAM (Character Generator RAM) içerisinde tanımlayabilir. CGRAM 'a kaydedilen veriler enerji kesilince silinir. Her enerji verildiğinde CGRAM'in tekrar yüklenmesi gerekir. Şekil 1.20 de en sol sütun CGRAM adreslerini göstermektedir. 0x00 ile 0x0F arası alan 16 alan boştur. 5x7 dot matris displaylerde 8 karakter tanımlanabilir. 5x10 dot matris displaylerde ise 4 karakter tanımlanabilir. İstenilen bir karakteri kaydetmek için önce CGRAM adresi seçilir, ardından karakter bilgisi gönderilir. Şekil 1.69 de uygulamada kullanacağımız özel tanımlı karakterler görünmektedir. Yanması istenen noktalar siyah renklidir. Nokta tespitini kullanıcı istediği şekilde belirler. En alt satır imleç içindir, bu yüzden boş bırakılmıştır. Oluşturulan karakterlerin satır değerleri toplanır. Toplamda kolaylık olması bakımından her sütunun üstüne bit değerleri yazılmıştır. İstenen karakteri oluşturmak satır değerleri tek tek LCD ye gönderilir. CGRAM adres kurma komutu şekildeki gibidir.

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Şekil 1.68: CGRAM adres kurma yazmacı

En baştaki Lojik 1 bilgisi sabittir. CGRAM adres kurma komutunun en düşük değeri 1000000 veya 0x40h tır. CGRAM adres kurma komutunun en yüksek değeri ise 1111111 veya 0x7F h' tir. LCD ye ilk olarak CGRAM adres seçme komutu, ardından satır değeri gönderilir. CGRAM içerisinde içinde toplam 8 karakter tanımlanabilir. 1. karakter CGRAM'in 0. adresinde, 2. karakter CGRAM'in 1. adresindedir, 8. karakter CGRAM'in 7. adresindedir. Örnek olarak CGRAM'in 0. adresine bir karakter yazmak için 8 adet komut, 8 adet satır verisi yollanır. CGRAM'in 0. adresi için komut değerleri; 0x40, 0x41, 0x42, 0x43, 0x44, 0x45,0x46, 0x47 şeklindedir. CGRAM'in 1. adresi için komut değerleri ise; 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D,0x4E, 0x4F şeklindedir. Böylece CGRAM'in 7. Adresine kadar komut değerleri artarak devam eder.



Şekil 1.69 Uygulamada kullanılacak özel tanımlı karakterler

Türkçe ve Özel Karakterleri gösterebilmek için **ozel_karakter_LCD.c** isimli driver dosyasının program kodları aşağıda görülmektedir. Daha önce oluşturduğumuz **bizim_LCD.c** dosyasına ilaveler yaparak bu dosyayı oluşturacağız. LCD bağlantı şeması **bizim_LCD.c** kullanılan ile aynıdır.

Örnek 1.5 : Türkçe ve Özel Karakterli LCD Sürücü Dosyası Tasarımı ve Uygulaması

```
  //***Türkçe ve Özel Karakterli LCD Sürücü Dosyası Tasarımı ve Uygulaması
  //********************* ozel_karakter_LCD.c  ************************
#define rs  pin_b6 // LCD'nin RS ucu RB6 pinine bağlı
#define e   pin_b7 // LCD'nin E ucu RB7 pinine bağlı

//****** LCD'ye Komut Gönderme Fonksiyonu **********
void lcd_komut(byte komut)
{
output_b(komut>>4);   // Komutun yüksek değerli 4 bitini gönder
output_low(rs);       // LCD komut almak için ayarlandı
delay_cycles(1);      // 1 komut saykılı bekle
output_high(e);       // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);      // 1 komut saykılı bekle
output_low(e);        // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);          // 5 msn gecikme veriliyor
output_b(komut&0x0F); // Komutun düşük değerli 4 bitini gönder
```

```
output_low(rs);      // LCD veri almak için ayarlandı
delay_cycles(1);     // 1 komut saykılı bekle
output_high(e);      // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);         // 5 msn gecikme veriliyor
}
/******* LCD'ye Veri Gönderme Fonksiyonu **********/
void lcd_veri(byte veri)
{
if (veri=='Ö') veri=0; // Karakter Ö ise CGRAM 0.adresteki bilgiyi gönder
if (veri=='İ') veri=1; // Karakter İ ise CGRAM 1.adresteki bilgiyi gönder
if (veri=='Ü') veri=2; // Karakter Ü ise CGRAM 2.adresteki bilgiyi gönder
if (veri=='Ç') veri=3; // Karakter Ç ise CGRAM 3.adresteki bilgiyi gönder
if (veri=='#') veri=4; // Karakter "TL" ise CGRAM 4.adresteki bilgiyi gönder
if (veri=='&') veri=5; // Karakter "mikro" ise CGRAM 5.adresteki bilgiyi gönder
if (veri=='*') veri=6; // Karakter "pi" ise CGRAM 6.adresteki bilgiyi gönder
if (veri=='>') veri=7; // Karakter "gülen adam" ise CGRAM 7.adresteki bilgiyi gönder

output_b(veri>>4);   // Verinin yüksek değerli 4 bitini gönder
output_high(rs);     // LCD veri almaya hazırlandı
delay_cycles(1);     // 1 komut saykılı gecikme
output_high(e);      // E ucu lojik-1 yapıldı
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yapıldı
delay_ms(1);         // 1 msn gecikme veriliyor

output_b(veri&0x0F); // Verinin düşük değerli 4 bitini gönder
output_high(rs);     // LCD veri almaya hazırlandı
delay_cycles(1);     // 1 komut saykılı gecikme
output_high(e);      // E ucu lojik-1 yapıldı
delay_cycles(1);     // 1 komut saykılı bekle
output_low(e);       // E ucu lojik-0 yapıldı
delay_ms(1);         // 1 msn gecikme veriliyor
}

/******* LCD İmlec Konumlandırma Fonksiyonu ********/
void imlec(byte satir, byte sutun)
{
if (satir==1)   // Eğer 1. satır seçili ise
lcd_komut(0x80|(sutun-1));
if (satir==2)      // Eğer 2. satır seçili ise
lcd_komut(0xC0|(sutun-1));

}

//********* LCD Başlangıç Ayarları Fonksiyonu ******
```

```
void lcd_hazirla()
{
int i=0;
output_low(rs); // RS ucu lojik-0
output_low(e);  // E ucu lojik-0
delay_ms(100);   // LCD enerjlendiğinde LCD'nin hazır olması için beklenen süre
for(i=0;i<3;i++)    // LCD'ye 3 kez 0x03 fonksiyon komutu gönderiliyor
{
lcd_komut(0x03);
delay_ms(10);      // 10 msn gecikme veriliyor
}
lcd_komut(0x02); // LCD'ye 4 bit iletişim komutu gönderiliyor
lcd_komut(0x28); // 4 bit iletişim, 2 satır, 5x8 dot matris seçildi
lcd_komut(0x08); // Display Kapalı, imleç gösterilmez ve yanıp sönmez
lcd_komut(0x01); // Display sil. İmleç 1.satır 1.sütunda
lcd_komut(0x06); // Her gönderilen karakterden sonra imleç bir sağa gitsin
lcd_komut(0x0C); // Display açık,imleç gösterilmez ve yanıp sönmez.
}

/***  CGRAM'e Özel Karakterleri Yükleyen Fonksiyon ****/
void ozel_karakterler()
{
// CGRAM 1. Adrese "Ö" Karakteri Kaydediliyor
lcd_komut(0x40);lcd_veri(10);
lcd_komut(0x41);lcd_veri(0);
lcd_komut(0x42);lcd_veri(14);
lcd_komut(0x43);lcd_veri(17);
lcd_komut(0x44);lcd_veri(17);
lcd_komut(0x45);lcd_veri(17);
lcd_komut(0x46);lcd_veri(14);
lcd_komut(0x47);lcd_veri(0);

// CGRAM 1. Adrese "İ" Karakteri Kaydediliyor
lcd_komut(0x48);lcd_veri(4);
lcd_komut(0x49);lcd_veri(0);
lcd_komut(0x4A);lcd_veri(4);
lcd_komut(0x4B);lcd_veri(4);
lcd_komut(0x4C);lcd_veri(4);
lcd_komut(0x4D);lcd_veri(4);
lcd_komut(0x4E);lcd_veri(4);
lcd_komut(0x4F);lcd_veri(0);

// CGRAM 2. Adrese "Ü" Karakteri Kaydediliyor
lcd_komut(0x50);lcd_veri(10);
lcd_komut(0x51);lcd_veri(0);
lcd_komut(0x52);lcd_veri(17);
lcd_komut(0x53);lcd_veri(17);
```

```
lcd_komut(0x54);lcd_veri(17);
lcd_komut(0x55);lcd_veri(17);
lcd_komut(0x56);lcd_veri(14);
lcd_komut(0x57);lcd_veri(0);

// CGRAM 3. Adrese "Ç" Karakteri Kaydediliyor
lcd_komut(0x58);lcd_veri(14);
lcd_komut(0x59);lcd_veri(17);
lcd_komut(0x5A);lcd_veri(16);
lcd_komut(0x5B);lcd_veri(16);
lcd_komut(0x5C);lcd_veri(16);
lcd_komut(0x5D);lcd_veri(21);
lcd_komut(0x5E);lcd_veri(14);
lcd_komut(0x5F);lcd_veri(0);

// CGRAM 4. Adrese "TL" Karakteri Kaydediliyor
lcd_komut(0x60);lcd_veri(10);
lcd_komut(0x61);lcd_veri(12);
lcd_komut(0x62);lcd_veri(26);
lcd_komut(0x63);lcd_veri(12);
lcd_komut(0x64);lcd_veri(25);
lcd_komut(0x65);lcd_veri(9);
lcd_komut(0x66);lcd_veri(14);
lcd_komut(0x67);lcd_veri(0);

// CGRAM 5. Adrese "mikro" Karakteri Kaydediliyor
lcd_komut(0x68);lcd_veri(17);
lcd_komut(0x69);lcd_veri(17);
lcd_komut(0x6A);lcd_veri(19);
lcd_komut(0x6B);lcd_veri(29);
lcd_komut(0x6C);lcd_veri(16);
lcd_komut(0x6D);lcd_veri(16);
lcd_komut(0x6E);lcd_veri(16);
lcd_komut(0x6F);lcd_veri(0);

// CGRAM 6. Adrese "pi" Karakteri Kaydediliyor
lcd_komut(0x70);lcd_veri(31);
lcd_komut(0x71);lcd_veri(10);
lcd_komut(0x72);lcd_veri(10);
lcd_komut(0x73);lcd_veri(10);
lcd_komut(0x74);lcd_veri(10);
lcd_komut(0x75);lcd_veri(10);
lcd_komut(0x76);lcd_veri(19);
lcd_komut(0x77);lcd_veri(0);
```

```
// CGRAM 7. Adrese "gülen adam" Karakteri Kaydediliyor
lcd_komut(0x78);lcd_veri(0);
lcd_komut(0x79);lcd_veri(2);
lcd_komut(0x7A);lcd_veri(17);
lcd_komut(0x7B);lcd_veri(13);
lcd_komut(0x7C);lcd_veri(17);
lcd_komut(0x7D);lcd_veri(2);
lcd_komut(0x7E);lcd_veri(0);
lcd_komut(0x7F);lcd_veri(0);
}
```

Bu programda önceden oluşturduğumuz **bizim_LCD.c** isimli programa ilaveler yapılmıştır. LCD ye veri gönderme fonksiyonununda **void lcd_veri(byte veri)** ifadesi ile **lcd_veri** isminde bir fonksiyon oluşturulmuş, **byte veri** ifadesi ile veri değişkeninin 1 bayttan oluştuğu belirtilmiştir**. if (veri=='Ö') veri=0;** ifadesi ile veri değeri Ö ise CGRAM 0.adresteki Ö bilgisinin LCD ye gönderilmesi sağlanmıştır. Diğer verilerin gönderilmesini sağlayan if komutları da aynı mantık çerçevesinde oluşturulmuştur.

void ozel_karakterler() ifadesi ile **ozel_karakterler** isimli bir fonksiyon oluşturularak CGRAM in adreslerine özel karakterler kaydedilmiştir. **lcd_komut(0x40);** komutu ile CGRAM 0. adrese kaydedilecek **Ö** karakterinin ilk satırının adres bilgisi LCD ye gönderilmiştir. Hemen ardından **lcd_veri(10);** komutu ile **Ö** karakterinin ilk satır bilgisi gönderilmiştir. Ö karakterine ait kalan diğer 7 satırın ilk adres ve satır bilgileri aynı mantık çerçevesinde LCD ye gönderilmiştir. Diğer tüm özel karakterler için aynı işlem tekrar edilmiştir. Programda Türk Lirası için # sembolü, mikro karakteri için & sembolü, Pi karakteri için * sembolü, gülen adam karakteri için > sembolü kullanılmıştır.

Oluşturduğumuz **ozel_karakter_LCD.c** dosyasını programlarımızda kullanabilmemiz için uzantısı ".C" olacak şekilde Drivers klasörü içine kaydetmemiz gerekmektedir. Bu klasörün yeri kurulum esnasında farklı bir yer seçilmediyse C:\Program Files\PICC\ Drivers şeklindedir.

Örnek 1.6: ozel_karakter_LCD.c Dosyası ile LCD Uygulaması Program Kodu:

```
/******Türkçe ve Özel Karakter LCD Sürücü Dosyası Uygulaması**********/
#include <16f877.h>     //denetleyicinin başlık dosyası tanıtılıyor.
#fuses  XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP, NOPUT,
NOWRT, NODEBUG,NOCPD // Denetleyici konfigürasyon ayarları yapılıyor

#use delay (clock=4000000) // Osilatör frekansı belirtiliyor.
#include <Ozel_karakter_LCD.c> // ozelkarakter_LCD.c dosyası programa ekleniyor.
float x=1.80;  // Ondalıklı tipte değişken tanımlanıyor
/********* ANA PROGRAM FONKSİYONU********/
void main ( )
{
setup_psp(PSP_DISABLED);        // PSP birimi devre dışı
setup_timer_1(T1_DISABLED);     // T1 zamanlayıcısı devre dışı
```

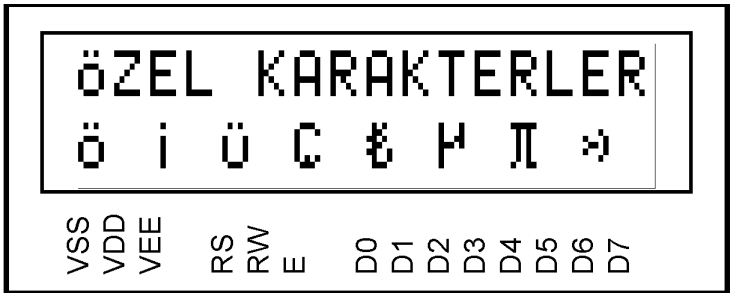
```
setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);    // ANALOG giriş yok
setup_adc(ADC_OFF);             // ADC birimi devre dışı
setup_CCP1(CCP_OFF);            // CCP1 birimi devre dışı
setup_CCP2(CCP_OFF);            // CCP2 birimi devre dışı

lcd_hazirla(); // LCD başlangıç ayarları yapılıyor
ozel_karakterler(); // Özel karakterler CGRAM'e yükleniyor

while(1) // Sonsuz döngü
{
imlec(1,1); // İmlec 1.satır 1.sütunda
printf(lcd_veri,"ÖZEL KARAKTERLERİMİZ"); // LCD'ye veri yazdırılıyor
imlec(2,1); // İmlec 2.satır 1.sütunda
printf(lcd_veri,"Ö İ Ü Ç # & * >");  // Özel karakterler yazdırılıyor.
delay_ms(2000); // 2 sn bekleme
lcd_komut(0x01); // LCD siliniyor
printf(lcd_veri,"İYİ GÜNLER >");    // "İYİ GÜNLER :-)" ifadesi yazdırılıyor
imlec(2,1); // İmlec 2.satır 1.sütunda
printf(lcd_veri,"1 DOLAR= %f#",x); // "1 DOLAR= 1,80 TL" ifadesi yazdırılıyor
delay_ms(2000); // 2 sn bekleme
lcd_komut(0x01); // LCD siliniyor
printf(lcd_veri,"KAPASİTE= 10&F "); // "KAPASİTE= 10µF" ifadesi yazdırılıyor
imlec(2,1); // İmlec 2.satır 1.sütunda
printf(lcd_veri,"DAİRE ÇEVRE=2*r "); // "DAİRE ÇEVRE=2 π r" ifadesi
yazdırılıyor
delay_ms(2000); // 2 sn bekleme
lcd_komut(0x01); // LCD siliniyor
}
}
```

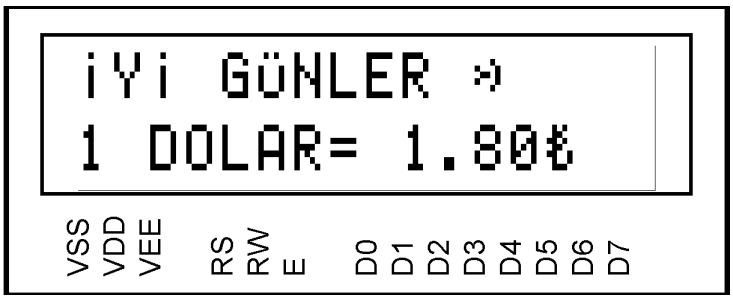
Açıklama: #include <Ozel_karakter_LCD.c> fonksiyonu ile **ozel_karakter_LCD.c** dosyası programa eklenmiştir. **lcd_hazirla();** komutu ile LCD nin ilk açılış ayarlarını yapan fonksiyon çağırılmıştır. **ozel_karakterler();** fonksiyonu ile özel karakterler CGRAM e yüklenmiştir. Daha önceden de belirtildiği gibi LCD nin enerjisi kesildiğinde CGRAM hafızası silinmektedir. Bu yüzden LCD nin ilk açılış ayarlarını yapan **lcd_hazirla();** fonksiyonundan hemen ardından **ozel_karakterler();** fonksiyonu çağırılarak CGRAM hafızası yüklenmelidir.

printf(lcd_veri,"ÖZEL KARAKTERLERİMİZ"); komutu ile **ÖZEL KARAKTERLERİMİZ** stringi 1. Satır 1. Sütuna yazılıyor. imlec(2,1); komutu ile imlec 2.satır 1.sütuna konumlandırılıyor. **printf(lcd_veri,"Ö İ Ü Ç # & * >");** komutu ile tanımlanan özel karakterler 2. satır . sütuna yazılıyor yazdırılıyor. Ekran çıktısı şekilde görülmektedir.



Şekil 1.70: Örnek 1.6 için 1. Ekran Çıktısı

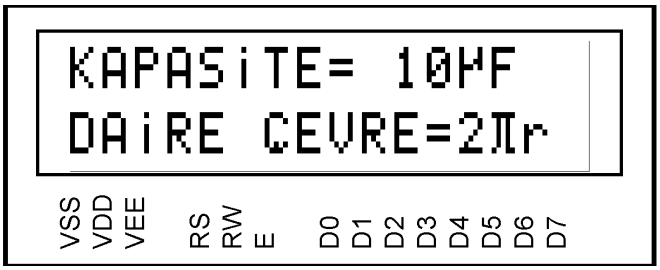
printf(lcd_veri,"İYİ GÜNLER >"); komutu ile "İYİ GÜNLER :-)" ifadesi yazdırılıyor. **printf(lcd_veri,"1 DOLAR= %f#",x);** komutu ile "1 DOLAR= 1,80 TL" ifadesi 2. satıra yazdırılıyor. Ekran çıktısı şekilde görülmektedir.



Şekil 1.71: Örnek 1.6 için 2. Ekran Çıktısı

printf(lcd_veri,"KAPASİTE= 10&F "); komutu ile "KAPASİTE= 10µF" ifadesi 1. satıra yazdırılıyor.

printf(lcd_veri,"DAİRE ÇEVRE=2*r "); komutu ile "DAİRE ÇEVRE=2 π r" ifadesi 2. satıra yazdırılıyor. Ekran çıktısı şekilde görülmektedir.

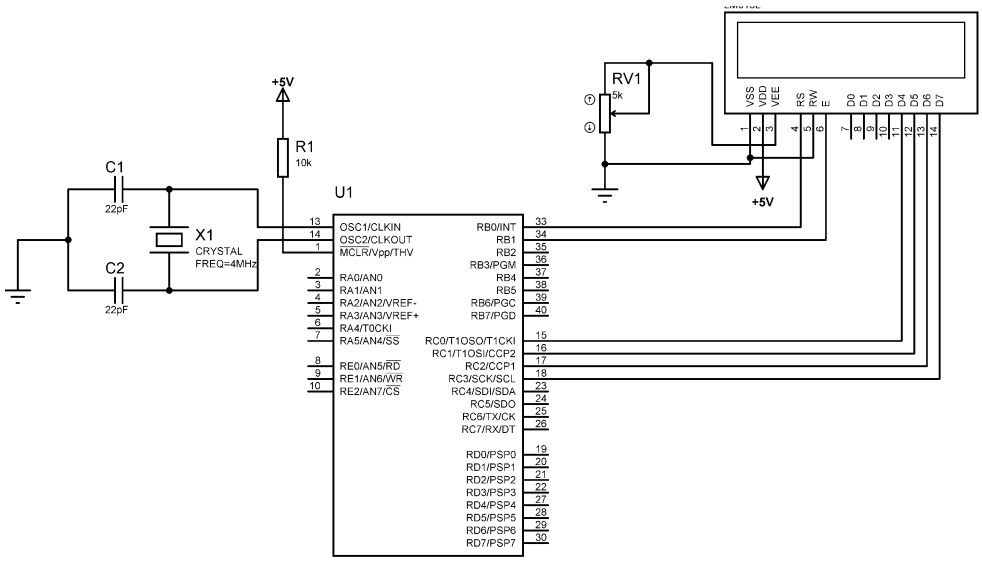


Şekil 1.72: Örnek 1.6 için 3. Ekran Çıktısı

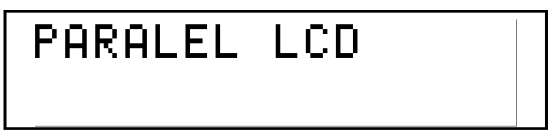
UYGULAMA FAALİYETİ

Kullanılan Malzeme ve Araç-Gereç

- 1 adet PIC16F877
- 1 adet 40 bacaklı entegre soketi
- 1 adet HD44780 işlemcili LCD
- 1 adet LCD için uygun 14'lü soket
- 1 adet 4 MHz Kristal
- 2 adet 22pF kondansatör
- 10KOhm direnç ve 5 KOhm trimpot
- 1 adet buTon
- PIC programlayıcı devresi
- 5V DC gerilim kaynağı
- 1 adet deney bordu
- Yeterli sayıda zil teli
- Yankeski ve kargaburun



Şekil 1.73: Uygulama Faaliyeti Devresi



Şekil 1.74: Uygulama Faaliyeti Ekran Çıktısı.

İşlem Basamakları	Öneriler
➢ Şekil 1.75'teki devre için gereken malzemeleri ve araç gereci hazırlayınız.	➢ 5V gerilim kaynağı kaliteli bir kaynak olmalıdır.
➢ PIC16F877'nin ve HD44780 LCD'nin katalog bilgilerini inceleyiniz.	➢ Entegrenin ve LCD'nin çalışmasını, karakteristik özelliklerini inceleyiniz
➢ Şekildeki bağlantıya uygun olarak LCD sürücü dosyası oluşturarak uyg1_LCDriver.c ismini vererek Driver klasörüne kaydediniz.	➢ bizim_LCD. c dosyasındaki pin tanımlamalarını değiştererek uyg1_LCDriver.c isminde kaydediniz.
➢ PARALEL LCD ifadesinin 1 sn aralıklarla sırası ile önce 1. Satırda, sonra 2. satırda gösterilmesini sağlayan programı yazarak Uyg_1.c ismini vererek kaydediniz.	➢ İlgili satırda ifade gösterildikten sonra LCD silme komutunu (01h) kullanınız.
➢ Yazdığınız programı derleyeyerek hex dosyasını oluşturunuz.	➢ Oluşturulan hex dosyayı ISIS programında test ediniz.
➢ Oluşturulan hex dosyayı denetleyiciye yükleyiniz, denetleyiciyi bord üzerine takarak öğretmeniniz gözetiminde devreyi test ediniz.	➢ Denetleyiciyi borda takıp çıkarırken pinlerin yamulup kırılmamasına özen gösteriniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. PIC16F877 mikrodenetleyicinin program belleği kaç KB'dır?
A) 8
B) 16
C) 32
D) 24

2. PIC16F877 mikrodenetleyicide kaç adet port bulunmaktadır?
A) 3
B) 4
C) 5
D) 6

3. PIC16F877'de bank seçimi için hangi yazmaç kullanılır?
A) Option
B) Intcon
C) Status
D) PIR1

4. LCD'de dışarıdaki elektronik birimlerle haberleşen, bu birimlerin gönderdiği kodları uygulayan birime ne ad verilir?
A) DDRAM
B) CGRAM
C) Mikroişlemci
D) Sıvı kristal ekran

5. LCD ekranında görüntülenecek karakterin kayıtlı olduğu bellek hangisidir?
A) CGROM
B) CGRAM
C) Mikroişlemci
D) DDRAM

6. Aşağıdakilerden hangisi ASCII tablosu verilerini saklar?
A) CGRAM
B) DDRAM
C) CGROM
D) DDROM

7. LCD'den veri okumak için aşağıdakilerden hangisi kesinlikle yapılmalıdır?
A) RS=0, E =1
B) RS=1, E=1
C) RS=0, R/W=0
D) RS=1, R/W=1

8. Özel karakterler LCD nin hangi hafızasına kaydedilir.
A) CGROM
B) DDRAM
C) Mikroişlemci
D) CGRAM

9. LCD.C sürücüsü kullanıldığında aşağıdakilerden hangisi LCD'yi ilk kullanıma hazırlar?
A) Kur()
B) Lcd_kur()
C) Lcd_init()
D) Dsply_kur()

10. Lcd_gotoxy(10,2) komutu ile LCD ekranında hangi fiziksel adrese ulaşılır?
A) 12
B) 49
C) 4A
D) CA

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Mikrodenetleyici ile servo motor kontrolünü hatasız olarak yapabileceksiniz.

ARAŞTIRMA

- Servo motor çeşitleri ve yapıları hakkında ön araştırma yapınız.

2. MİKRODENETLEYİCİ İLE SERVO MOTOR KONTROLÜ

Otomasyon sistemlerinde üretimin otomatik olarak gerçekleşmesi için elektrik enerjisinin harekete çevrilmesi gerekmektedir. Hareket elde etmek için kullanılan en önemli elemanlar motorlardır. Bu öğrenme faaliyetine başlamadan önce bu motorların yapısını incelemenizde fayda vardır. Doğru akım motoru ve adım motoru ile hassas açılara dönme hareketi yapmak zordur. Özellikle robot sistemlerinde belirli açılarla dönme hareketi yapılmalıdır. Bu nedenle doğru akım motorları geliştirilmiş ve "servo motor" ortaya çıkmıştır. Servo-motorlar, verilen girişe göre istenen açısal konuma gelen motorlardır. Günümüzde ac veya dc gerilimle çalışabilen, çeşitli güçlerde ve ebatlarda servo motorla mevcuttur. Servo motorların çok çeşitli uygulamalarda kullanılmasını nedenleri;

- Güvenilir olması
- Yüksek tork değerlerine sahip olması
- Konumlamada yüksek doğruluk göstermesi
- Kolay kurulum ve bağlantı özelliği
- Kontrol kolaylığı
- Ekonomik oluşu olarak verilebilir.

Bu öğrenme faaliyetinde literatürde "RF Servo Motor" ya da "RC Servo Motor" olarak da geçen DC Servo Motorun temel yapısı incelenecektir.

2.1. Servo Motorun Yapısı

Sahip olduğu bir mil üzerinden gücünü dışarıya aktarabilen mekanizmaya "Servo" adı verilir. Bu mekanizmaya özel olarak tasarlanmış motor birimi ve dişli kutusunun ilave edilmesiyle servo motorlar üretilirler. Hobi amaçlı kullanılan küçük çaplı robotlarda ve oyuncak arabalarda kullanılan servo motorlar için "RF(RC) Servo Motor" kavramı kullanılır. Uzaktan kumandalı uçak, araba vs gibi hobi araçlarının yönlendirme mekanizmalarının harekete geçirilmesi, radyo kontrollü gemiler ve kuklalar diğer kullanım alanlarının başında gelir. Mil hareket edebildiği aralıkta belirli bir pozisyona, servoya

gönderilen kodlu bir sinyale uygun olarak getirilebilir. Kodlu sinyal servonun giriş ucunda bulunduğu sürece konum aynen korunacaktır. Sinyaldeki değişimlere göre milin açısı da değişecektir.

Şekil 2.1: Servo motorlar

Servo motor robot kontrolünde son derece kullanılışlı bir motordur. Şekil 2.1'de görüldüğü gibi küçük boyutlu olmasına rağmen çok güçlüdür ve çok az akım çeker. Bunun yanı sıra yüksek döndürme momentlerine(tork) sahiptirler. Servo-motorlar 4,7 ile 7 Volt besleme gerilimleri arasında çalışabilirler, kontrol sinyali gerilimi TTL seviyesinde olmalıdır. Servo motorlar yüksek güç ve uygun sinyal tepkileri için genellikle 5V'a yakın gerilimlerde çalıştırılır.

Şekil 2.2'de servo motor iç yapısında bulunan parçalar görülmektedir. Elektronik kumanda devresi, geri besleme potansiyometresi, dc motor, dişli grubu ve dişli kutusu servo motoru meydana getirir. DC motor çift mıknatıslı bir stratora ve fırçalı rotora sahiptir. Motor mili belirli bir dönme oranına sahip dişli sistemine bağlıdır. Böylece yüksek bir tork değerine ulaşılır. Dişli sisteminin çıkışında geri besleme potansiyometresi mil konumunu elektronik kumanda devresine iletir. Kumanda devresinin görevi, mil konumunun Ton süresi ile ifade edilen açıya gelmesi için motoru döndürmek ve açı noktasında durdurmaktadır.

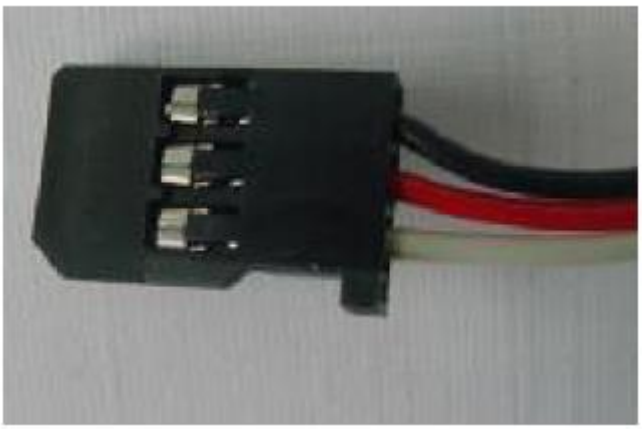
Dışarıdaki kontrol devresinden teorik olarak 1 ile 2 milisaniye arasında Ton süresi değişen bir kontrol sinyali her 20 milisaniyede bir servoya gönderilir. Ton süresi 1,5 ms olduğunda servo motor mili başlangıç pozisyonundadır. Ton süresi 1 milisaniye ise motor mili başlangıç pozisyonuna göre -90 derece bir yöne, 2 milisaniye ise +90 derece zıt yöne konumlanır. 1-2 ms arasında herhangibir zamanlama ile de -90 ve +90 derecelik açıların arasındaki herhangibir açıya konumlanabilir. Ancak bu açı aralığı pratikte 1-2 ms olmayabilir. Servo kontrol devresi tasarımı için öncelikle kataloglara başvurulmalı ve motor milinin konumlanacağı yön ve süreler denenmelidir. Uygun zamanlama sinyalleri servoya uygulandığında, içerideki elektronik kumanda devresi, ilk önce gelen darbelerin darbe genişliğini ölçer, daha sonra geri besleme potansiyometresinin konumuna bakar ve kendi darbe osilatörünün darbe genişliği gelen darbelerle eşitleninceye kadar motoru hareket ettirir.

Servo motor içerisinde yer alan elektronik kumanda devresi ve geri besleme potansiyometresi çok önemli yapılardır. Şekil 2.2'de potansiyometre motorun hemen yanında görülmektedir. Geri besleme potansiyometresi lineer değişen kaliteli bir potansiyometredir ve mile doğrudan bağlıdır. Mille birlikte dönme hareketi yapar. Böylece direnç değeri değişir. Potansiyometrenin kullanım amacı milin pozisyonunun doğru değerde olup olmadığını anlamaktır. Mil gereken açıya gelmişse potansiyometre bunu algılar ve motor durur. Kontrol devresine gelen dönüş açısı bilgisi yanlış ise motor doğru açıyı bulana kadar döndürülür.

Servomotor milinin taradığı açı teorik olarak maksimum 180°'dir. Bazı servo motor modellerinde maksimum tarama açısı 210° ye kadar artabilmektedir. Maksimuım tarama açısı kataloglardan ve deneme yanılma yoluyla bulunabilir. Servo motorların iç yapısında yer alan mekanik sınırlama nedeniyle servo motor mili bu süpürme aralığının dışına çıkamaz.

Şekil 2.2: Servo motor iç yapısı

Şekil 2.3'te 3 adet kablo görülmektedir. Bunlar servo motorun dışarıdaki elektronik sistemlerle bağlantısını sağlarlar. Bunlardan kırmızı olan +5V gerilim için, siyah olan şase bağlantısı için ve yeşil olan da servo kontrol siyali için kullanılır. Bazı servolarda kontrol sinyali için kullanılan kablonun rengi beyaz veya turuncu olabilmektedir.

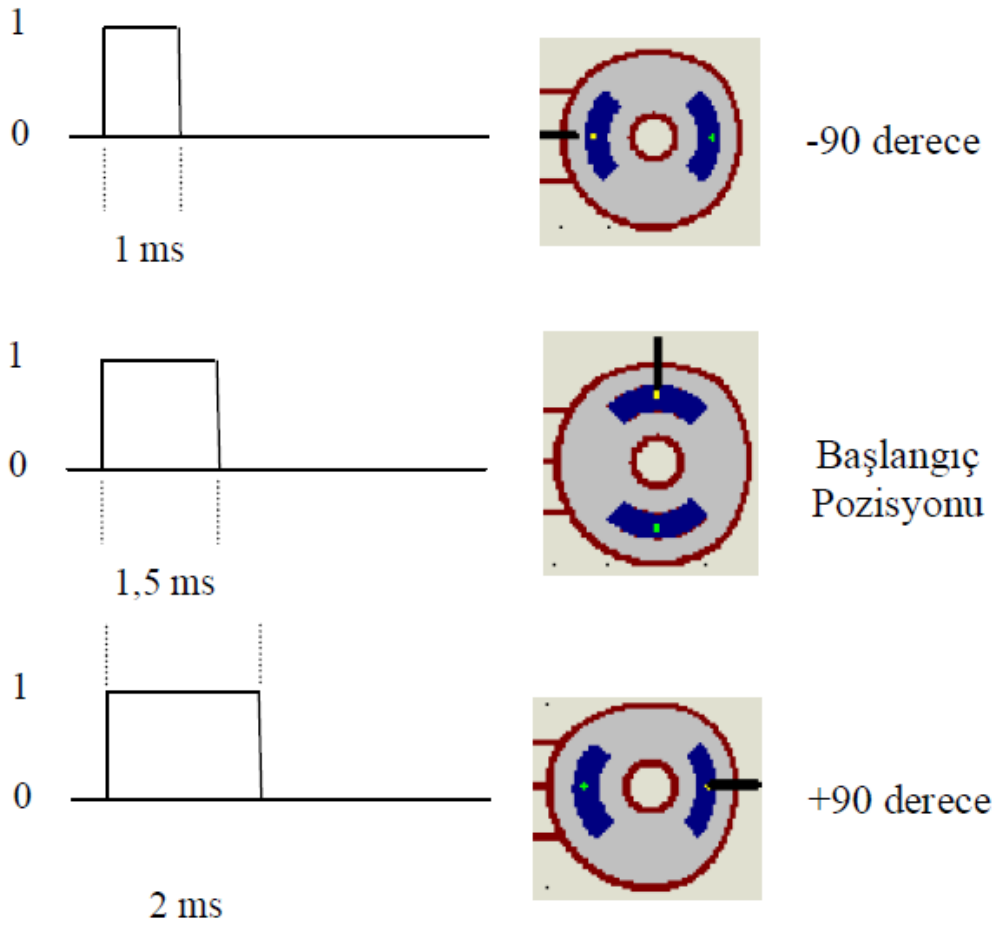


Şekil 2.3: Servo motor bağlantı kabloları

Elektronik kumanda devresince motora uygulanan güç katedilmesi gereken mesafe ile doğru orantılıdır. Eğer motor milinin geniş bir açıyı katetmesi gerekiyorsa motor tam hızda çalıştırılır. Eğer küçük bir açıda dönme gerekiyorsa motor hızı kumanda devresince düşürülür. Bu çalışma için kumanda devresinin kullandığı kontrol yöntemi oransal kontrol yöntemidir.

Oransal kontrolün gerçekleşmesi için bir kontrol sinyaline ihtiyaç vardır. Bu control sinyali daha önce anlatıldığı gibi servo motora milin konumlanması gereken açıyı bildirir. Dışarıda mikroişlemci tabanlı bir sistem örneğin PIC16F877 servo motorun kontrol bilgisini çok rahat üretebilir. Milin konumlanacağı açı, bu bağlantı üzerinden gönderilen bir sinyalin lojik-1 seviyesinde kalma süresi (Ton) ile belirlenir. Ton süresi "darbe genişlik modülasyonu (PWM)" ile ayarlanır.

Yatay eksen referans alındığında, servo motor milinin 90°'lik açı pozisyonuna nötr pozisyon da denilebilir ve açı zamanlaması hesabında 0 noktası olarak alınır. Nötr pozisyonun solunda kalan açılar saat yönünün tersine artacak şekilde negatif, sağında kalan açılar da saat yönünde artacak şekilde pozitif olarak düşünülebilir. Nötr pozisyon için gereken Ton süresinin 1,5 ms olduğu varsayılırsa, eğer Ton süresi 1.5 ms den kısa süreli ise mil konumu -90° ye doğru, 1.5ms den uzun ise mil +90° ye doğru ilerleyecektir. Ton süresi 1 ms ise servo mili başlangıç pozisyonuna göre -90 dereceye, 2 ms ise +90 dereceye konumlanır. Bu düşünce şekli servo motor milinin konumlanacağı açıları anlayabilmek açısından tasarımcıya yarar sağlar. Bu anlatılanlardan farklı düşünce şekilleri de üretilebilir. Şekil 2.4'te örnek olarak bir servo motorun kontrol sinyallerine göre ulaştığı konumları göstermektedir.



Şekil 2.4: Kontrol sinyallerinin servo motora etkisi

Şekil 2.4'te görüldüğü gibi Ton süresi milin konumlanacağı açıyı belirler. Ton süreleri ve mil sapma yönleri üreticiden üreticiye farklılık gösterebilir. Ancak servo motorların zamanlama prensipleri aynıdır. Servo motorlu bir sistem tasarlanmak istendiğinde catalog bilgilerine başvurulmalı ve servo karakteristikleri ön uygulamalarla belirlenmelidir.

2.2. Servo Motorun Mikrodenetleyici ile Kontrolü

Servo motoru kontrol etmek için besleme gerilimi ve bir kontrol sinyalinin gerektiğini daha önce belirtmiştik. Servo motoru kontrol etmek için elektronik bir devre tasarlanmalıdır. Bu elektronik devrenin mikrodenetleyici tabanlı olmasında fayda vardır. Bir servo motoru sürmek için tasarlanan elektronik devrenin güç kısmı "servo sürücü" olarak adlandırılır. Endüstriyel robotlarda ve diğer nümerik olarak kontrol edilen ekipmanlarda servo sürücüler geniş ölçüde kullanılır. Servo sürücünün kullanım amacı yüksek güç gerektiren servo motorlara gerekli akım ve gerilimi sağlamaktır. Servo sürücüye servo motorun hareketi için uygun kontrol sinyalleri uygulanır. PWM sinyali uygulanan servo sürücülere PWM servo

sürücüler adı da verilir. PWM modülasyonu ile transistör veya mosfetler anahtarlanarak servo motora daha yüksek güç aktarılabilir.

Transistorlü devrelerde, özellikle transistorlerin lineer çalışma bölgesinde ısı şeklinde ortaya çıkan bir güç harcanır. Bu nedenle güç kaybı meydana gelir. Transistörlerdeki güç kaybını azaltmak ve servo sürücülerin verimini iyileştirrnek için DC konvertörlerde olduğu gibi darbe genişlik modulasyon tekniği kullanılır. Bir PWM sürücüde, transistör ya doyum bölgesinde yada kesim bölgesindedir. Böylece kaybedilen güç azalır ve servo motora daha yüksek güç aktarılır.

Servo motor kontrol sinyalinin uygulanması ile hareket ettiğinde hareketini gerçekleştirmek için besleme kaynağından akım çeker. Kontrol devresi ve servo motor aynı besleme kaynağından akım çektiğinde istenmeyen parazit sinyaller ve gerilim dalgalanmaları oluşabilir. Bu durum eğer güç kaynağı kalitesiz ise mikrodenetleyicinin çalışmasını etkiler.

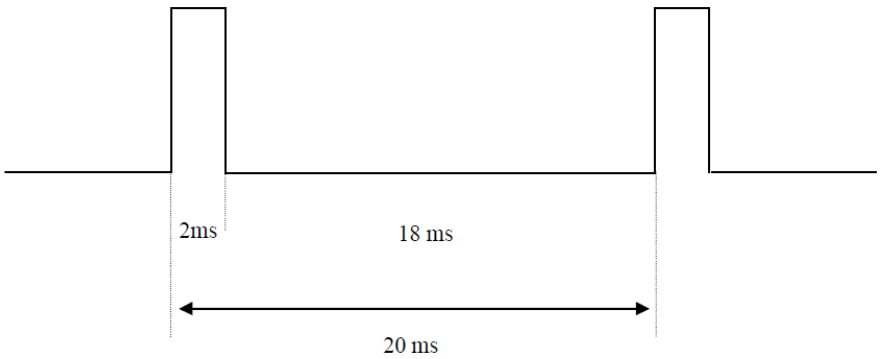
Düşük güçlü servo motorlar mikrodenetleyici portuna bir led diyot gibi doğrudan bağlanabilirler. Ancak karmaşık sistem tasarımnlarında servomotorun mikrodenetleyici ile aynı kaynağı kullanması özellikle motorun dönüş hareketleri sırasında program çalışmasını etkileyebilir.

Servomotorun gücünü arttırmak için kondansatör kullanılabilir. Kondansatör deşarj akımı servo motora takviye yaparak gücünü arttırır. En etkin çözüm servo motor ile mikrodenetleyici arasında tampon bir eleman kullanmaktır. Örneğin şekil 2.7'da görüldüğü gibi mosfet kullanılarak denetleyici ve motorun besleme kaynakları ayrılabilir.

Sürücü tasarımında diğer önemli bir nokta da servomotorların boyutları ve özelliklerinin farklı olabileceği gerçeğidir. Bu bağlamda kataloglara başvurularak karakteristik özelliklerine ve kablo renklerine dikkat etmek gerekir.

Yazılan programda ya da sayısal çıkış üreten devrede servonun kontrol girişine uygun bir kare dalga sinyal tasarlanır. Eğer bir mikrodenetleyici programı yazılıyorsa +90 ve -90 derecelik açılar için gereken Ton zaman değerleri arasında bir lojik-1 süresi ayarlanmalıdır. Her lojik-1 darbesi arasında 10-18 ms lojik-0 ile bekleme yapılmalıdır. Aslında her Ton süresi arasında 10 ms'lik Toff süreleri servo motorun hareket etmesi için yeterli olacaktır.

Servo motora kontrol sinyali uygulanmazsa mil boşta kalır ve el ile bile döndürülebilir. Servonun konumunu koruması için sürekli olarak kontrol sinyalinin uygulanması gerekir. Ton süresi değiştiğinde servo motor mili yeni açı konumuna hareket eder. Aşağıda periyotu 20 ms olan bir kontrol sinyali görülmektedir.



Şekil 2.5: Örnek kontrol sinyali

Şekil 2.5'te servo motor milini teorik olarak +90 derecelik konuma getiren control sinyali görülmektedir. Sinyal şeklinde görüldüğü gibi Ton darbesi 18 ms aralıklarla sürekli verilmektedir. Süreklilik sağlanmazsa mil boşta kalır. Eğer mile bağlı bir robot kolu varsa ağırlık nedeniyle mil konum değiştirebilir ve istenmeyen durumlarla karşılaşılabilir.

Daha önce belirtildiği gibi servo motor mili tam tur atamaz. Servo motorun içerisinde mekanik olarak sınırlama yapılmıştır. Servo motoru şekil 2.5'te görüldüğü gibi 2 ms Ton süreli sinyaller ve 10-18 ms'lik bekleme aralıklarını tekrarlayarak konumlayabiliriz. Daha sonra servo motoru bir uç konumdan diğer uç konuma tam tersi yönde konumlandırmak için de 1 msn. sinyal ve 10-18 ms'lik bekleme sürelerinden oluşan bir dizi sinyal gerekir.

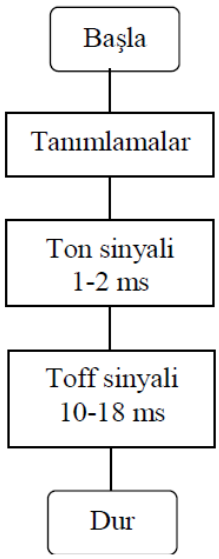
Kontrol sinyali arka arkaya uygulandığı zaman şaftın yeni konumuna ulaşması için bir süre geçer. Servo motorun iç devresinin kontrol sinyaline karşı olan tepki süresi de önemlidir. Sinyali üreten devrenin tasarımında tepki süresi ve yeni konuma ulaşma süresi de göz önüne alınmalıdır.

2.2.1. Servo Motorun Yazılım ile Kontrolü

PIC16F877 mikrodenetleyici kullanarak çok kaliteli zamanlama sinyalleri elde edilebilir ve servo motor hassas açılarla hareket ettirilebilir. Bu bağlamda program içerisinde PIC-C dilinin avantajarını kullanmak gerekir. Örneğin 1 ms'lik bir zamanlama için "delay_ms(1)" komutunu kullanmak yeterlidir.

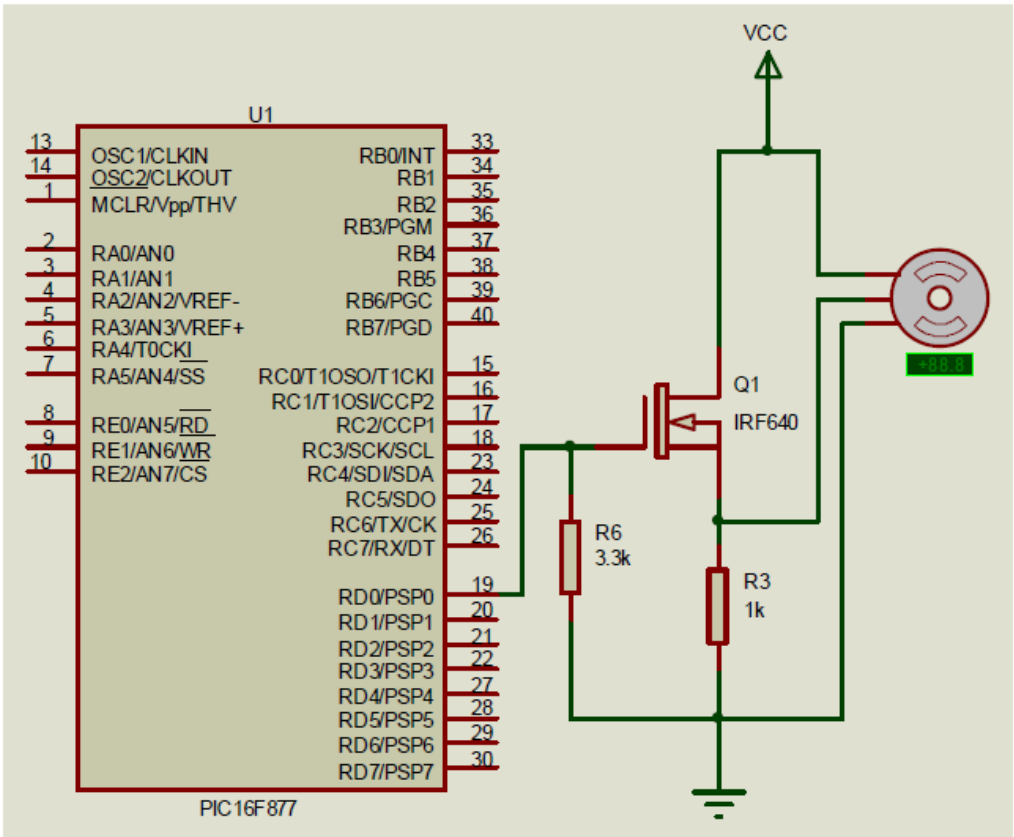
Şekil 2.6'de servo motoru kontrol etmek için gereken akış diyagramı görülmektedir. PIC16F877'yi ilk kullanıma hazırladıktan sonra servonun konumlanacağı açıyı bildiren zamanlama sinyali oluşturulur. Bu sinyal alt program yardımıyla da oluşturulabilir.

Öncelikle servo motorun kontrol ucuna bağlı port biti lojik-1 yapılır. Kullanılan servo motorun karakteristiğine göre Ton süresi ayarlanır ve bu süre kadar bekleme yapılır. Böylece kontrol ucuna açı kontrol bilgisi ulaşmış olur. Lojik-1 seviyesindeki beklemeden sonra PIC ucu lojik-0 yapılır. Aynı şekilde 10-18 ms lojik-0'da bekleme yapılır. Bu işlemler sürekli tekrarlanırsa motor mili boşta kalmadan sürekli açı pozisyonunu korur.



Şekil 2.6: Sürücü sinyali akış diyagramı

Şekil 2.7'da PIC16F877'li bir servo sürücü devresi örneği görülmektedir. Örnek devrede görüldüğü gibi PIC ile servo motor arasına n kanal bir mosfet bağlanmıştır. Servo motor dönüş hareketi yapmaya başladığında gerilim kaynağından akım çeker. Akım çekilmesi esnasında gerilim kaynağındaki dalgalanma mikrodenetleyiciyi etkiler. Mosfet kullanılmadan da 5V'luk servo motorlar doğrudan mikrodenetleyici portuna bağlanarak dönme hareketi yapabilir. Ancak mosfet gibi arada bir tampon eleman kullanıp servo motor için ikinci bir besleme kaynağı kullanmak sürücü devrenin ve tüm sistemin çok daha kaliteli ve güvenilir olmasını sağlayacaktır. Özellikle servo motor bir yükü hareket ettirdiği zaman daha fazla akıma ve dolayısıyla güce ihtiyaç duyacağından harici besleme kaynağı yararlı olacaktır. Harici besleme kaynağının kullanılıp kullanılmayacağı yapılan sisteme göre tasarımcının karar vereceği bir problemdir. Şekil 2.7'da mosfet ve servo motor için ikinci besleme kaynağı(Vcc) kullanılmıştır.



Şekil 2.7 : PIC16F877'li bir servo sürücü devresi

Şekil 2.7'daki devrede mosfet kullanıldığından ve çıkış kaynak (source) ucundan alındığından zamanlama sinyalleri mikrodenetleyici çıkışı ile aynı fazlı olarak üretilmiştir.

Örnek 2.1 de basit bir servo motor programı görülmektedir.

Örnek 2.1: Basit bir servo motor programı.

```
//SERVO MOTOR KONTROL PROGRAMI
//PD0: Servo kontrol sinyali için kullanılacak
#include <16F877.h>
#use delay(clock=20000000)
#fuses HS,NOWDT,NOPROTECT,NOLVP
#byte port_d=8 //portd'nin adresi
#define CNT PIN_D0
io_setout( )
{ set_tris_d(0x00); // port_d = çıkış
}
void main()
```

```
{io_setout();
while(1)
{
Output_High(CNT);
delay_us(1750); // Lojik-1 süresi 1,75 ms
Output_Low(CNT);
delay_us(18250); // Lojik-0 süresi 18,25 ms
}
} // Periyot 20 ms
```

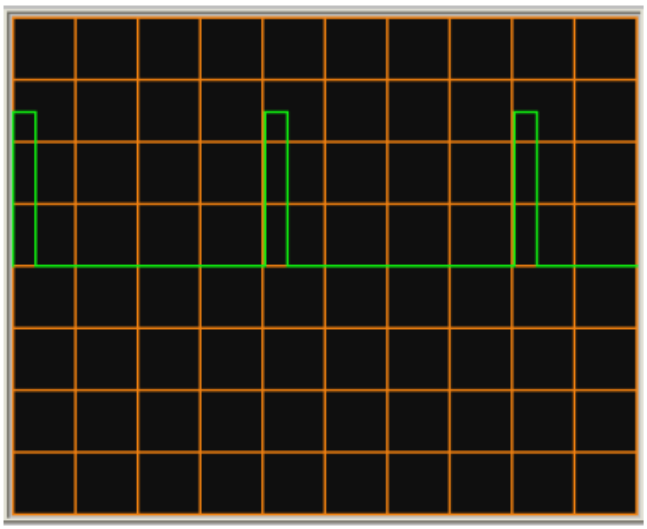
Şekil 2.8'da N kanal mosfet çıkışı görülmektedir. Mosfet çıkışı ile mikrodenetleyici çıkışı aynı fazlıdır. Mosfet kapı girişine lojik-1 seviyeli sinyal geldiğinde mosfet iletime geçer. Kaynaktaki R3 direncinin üst ucu lojik-1 olur. Böylece giriş lojik-1 iken çıkış da lojik-1'dir. Tam tersi bir şekilde kapı girişine lojik-0 seviyeli sinyal geldiğinde mosfet yalıtımdadır. Bu durumda mosfet yalıtımda olduğu için R3 direncinin üst ucu şase potansiyelindedir, diğer bir deyişle lojik-0'dır. Sonuç olarak girişdeki zamanlama sinyalinin aynısı çıkışta elde edilir. Mosfet sayesinde servo motor kendisi için gereken akımı gerilim kaynağından çeker.

Mosfet çıkışındaki sinyalin Ton süresi 1,75 ms, Toff süresi de 18,25 ms'dir. Mosfet çıkışındaki sinyalin periyotu ise;

T=Ton+Toff formülüne göre

T=1,75+18,25=20 ms bulunur.

Zamanlama sinyaline göre servo motor 1,75 ms ile ifade edilen açıya doğru dönüş yapar ve bu açıda bekler.



Şekil 2.8: Servo sürücü devresi mosfet çıkışı

Time/div=5 ms Volt/div=2V

Soru: Servo motor milinin nötr pozisyonu 1,5 ms, + 90 derece için Ton süresi 2 ms ve - 90 derece için Ton süresi 1 ms ise 1,75 ms'lik Ton süresi yaklaşık kaç derecelik açıya karşılık gelir?

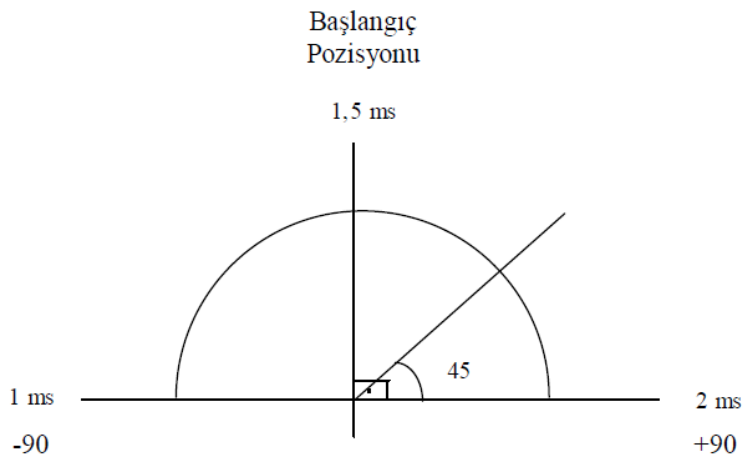
Soruda verilen karakteristiğe sahip servo motorda 1 derecelik dönüş için gereken Ton süresini yaklaşık olarak bulabiliriz. Buna göre;

Başlangıç pozisyonuna 0 derece dersek, 0 dereceden +90 dereceye kadar olan zaman farkı= 2ms-1,5ms=0,5 ms (500 µs) bulunur.

Derece başına düşen zaman=500 µs /90=5,55µs 'dir.

Servo motorun +45 derecelik konuma gitmesini istiyoruz. +45 derece için gereken Ton zaman farkını bulmak için;

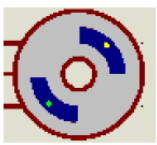
Ton=45.(5,55µs)=250 µs bulunur.



Şekil 2.9: Servo motor açı ve zaman durumları

Bulunan zaman farkı programda belirlenecek Ton süresi için kullanılacaktır. Motor mili 1,5 ms'lik başlangıç konumundan saat yönünde + 45 derece ilerleyecekse 1,5 ms'nin üzerine 0,25 ms eklenir. Örnek 2.1 programında 45 derecelik açı için gereken zamanlama tasarlanmıştır. 45 derece için gereken Ton süresi

Ton=1,5ms+0,25 ms=1,75 ms bulunur. Şekil 2.9'daki mosfet çıkışındaki sinyalin Ton süresinin 1,75 ms olması bu yüzdendir.



Şekil 2.10: Servo motorun 45 dereceye konumlanması

2.2.2. Servo Motorun Donanım PWM ile Kontrolü

Örnek 2.1'de yazılım yolu ile PWM sinyali elde edilmiştir. Bu programlar dikkatli bir şekilde incelenirse, özellikle mikrodenetleyici ile birden fazla iş yapılması durumu göz önüne alındığında mikrodenetleyicinin sürekli meşgul durumda olduğu ve ikinci bir iş yapıldığında pwm sinyalinin bundan etkileneceği anlaşılır. PWM üretiminin de yer aldığı kompleks sistemlerde mikrodenetleyicilerin çok fonksiyonluluk özelliğinden etkili bir şekilde yararlanılırsa, sistemin verimliliği ve güvenirliği artarken karmaşıklığı ve maliyeti azalır. Bu bağlamda, PIC16F877 mikrodenetleyici iç yapısında birçok elektronik modül vardır ve konumuzla ilgili olduğundan burada PWM modülünün kullanılmasına değinilecektir. PWM modülü sayesinde ana programın çalışmasını meşgul etmeden otomatik olarak bağımsız bir hattan PWM sinyali elde edilebilir. PIC-C dili ile PWM sinyali üretmek hazır fonksiyonlar sayesinde oldukça kolaylaştırılmıştır. Bu konuya girmeden önce PWM için kullanılan yazmaçlara göz atalım.

2.2.2.1. PWM için Kullanılan Yazmaçlar

PIC16F877 mikrodenetleyici ile PWM sinyali üretmek için TMR2, PR2, CCPR1 CCPR2, CCP1CON ve CCP2CON yazmaçları kullanılır. Bu kısımda ilgili yazmaçlar PWM sinyalinin üretimine yönelik olarak sunulacaktır.

CCPR1 ve CCPR2

CCPR1 ve CCPR2 yazmaçları 16 bit yakalama, 16 bit karşılaştırma ve PWM aktarma oranı belirleme görevlerini gerçekleştirir. CCPR1 ve CCPR2 yazmaçları aynı şekilde çalışırlar. Birbirinden bağımsız iki ayrı aktarma oranı bu yazmaçlarla elde edilebilir. CCPR1 ve CCPR2 yazmaçları CCPR1L, CCPR1H ve CCPR2L, CCPR2H olmak üzere iki adet 8 bitlik yazmaçtan oluşur.

Not: CCPR1(2) ifadesi CCPR1 ve CCPR2 yazmaçlarını ifade etmektedir.

CCP1CON ve CCP2CON

Bu yazmaçlar CCPR1(2) yazmaçlarını kontrol ederler. Yazmaçtaki bitler ve görevleri aşağıda verilmiştir.

7	6	5	4	3	2	1	0
-	-	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
		R/W	R/W	R/W	R/W	R/W	R/W

Şekil 2.11: CCP1CON1 ve CCP2CON Yazmaçları

Bit 7-6: Kullanılmayan bitlerdir.
Bit 5-4: Sadece PWM için kullanılırlar. PWM aktarma saykılının en düşük iki bitidir. Geriye kalan en büyük 8 bit CCPR1(2)L yazmacında bulunur. Aktarma saykılı PWM sinyalinin Ton olduğu süredir.
Bit 3-0: CCPR1(2) mod seçim bitleridir.

0000: PWM sinyali iptal
11xx : PWM moduna geçilir.

PR2

Timer2'nin periyot değeri 8 bitlik PR2 yazmaçına yazılır. Bu yazmaç yazılabilen ve okunabilen bir yazmaçtır. Mikrodenetleyici resetlendiğinde PR2'nin aldığı değer FF'dir. Timer2 zamanlayıcısı kullanılacaksa PR2 yazmacına 00h ile FFh arasında bir değer yazılmalıdır. Yazılan bu değere göre PWM sinyalinin periyotu ve frekansı belirlenir. Ayrıca PWM sinyalinin Ton süresinin belirlenmesinde de doğrudan etkilidir.

TMR2

Timer2 zamanlayıcısı 8 bitlik ön bölücü ve son bölücü değerlerine sahip bir modüldür. CCP modülü PWM amaçlı kullanıldığında zaman değerlerine doğrudan etki eder. Yazılabilir ve okunabilir bir yazmaçtır. Mikrodenetleyicinin resetlenmesi ile sıfırlanır. Timer2 modülünün zamanlama süresinde T2CON yazmacı etkindir. Bu yazmacın kontrolünde, giriş osilatör frekansı (Fosc/4) Timer2 için 1:1, 1:4 veya 1:16 ön bölücü seçeneklerinden biri ile işleme sokulur.

Timer2 modülü TMR2 ve daha önce bahsedilen PR2 yazmacını içerir. TMR2'de sayma ya da zamanlama değeri, PR2'de ise daha önce belirtildiği gibi zamanlama periyodunu saklanır. TMR2'nin değeri sayma(zamanlama) darbeleri ile 00h'den başlar PR2 değerine ulaşır. PR2 değerinden sonra 00h değerine döner. Zamanlama ya da sayma değeri TMR2 içeriği okunarak anlaşılabilir.

Timer2 modülünün iki çıkışı vardır. Bunlardan birincisi TMR2 çıkışı, diğeri de TMR2 kesme çıkışıdır. Kesme çıkışı için timer2 modülü 4 bitlik son bölücü değerine sahiptir. Son bölücü değeri 1:1'den 1:16'ya kadar olabilir. TMR2 kesmesi ile PIR1 yazmacının TMR2IF biti lojik-1 değerini alarak etkilenir.

Ön bölücü ve son bölücü işlemleri aslında Timer2 modülündeki bu iş için özelleştirilmiş sayıcılarla gerçekleştirilir. Bu sayıcılar aşağıdaki olaylar gerçekleştiğinde sıfırlanır.

- TMR2 yazmacına yazma
- T2CON yazmacına yazma
- POR, MCLR, WDT Reset ve BOR gibi herhangi bir resetleme işleminde

TMR2 yazmacının değeri aynı zamanda SSP modülü ile haberleşme sırasında kaydırma darbesi için kullanılır. Aşağıda ön bölücü ve son bölücü değerlerinin daha iyi kavranması için T2CON yazmacı verilmiştir. Timer2 modülü T2CON.2 biti sıfırlanarak kapatılabilir ve böylece güç harcaması azalabilir.

T2CON

7	6	5	4	3	2	1	0
-	Toutps3	Toutps2	Toutps1	Toutps0	Tmr2on	T2Ckps1	T2Ckps0
		R/W	R/W	R/W	R/W	R/W	R/W

Şekil 2.12: T2CON Yazmaçı

Bit 7: Kullanılmayan bittir.

Bit 6-3: Timer2 çıkış son bölücü seçim bitleridir.
b6 b5 b4 b3
0 0 0 0 = 1:1 son bölücü
0 0 0 1 = 1:2 son bölücü
:…:…:…:……:…
1 1 1 1 = 1:16 son bölücü

Bit 2: Timer2'nin çalışmasını başlatır.
1: Timer2 başlar.
0 : Timer2 durur.

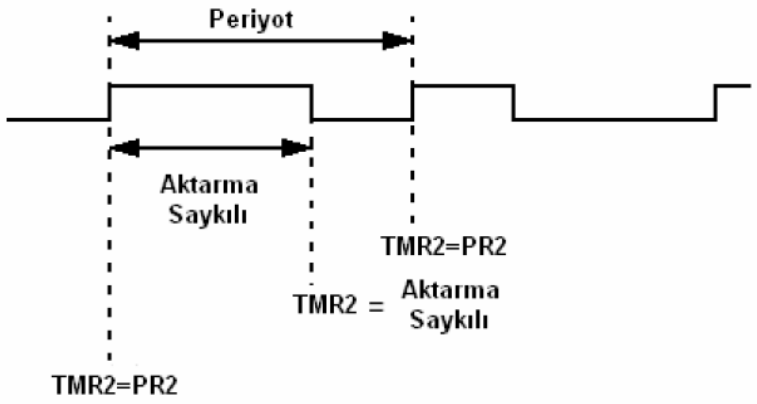
Bit 1-0: Timer2 için osilatör ön bölücü seçim bitleridir.
00: Ön bölücü değeri 1
01: Ön bölücü değeri 4
1x: Ön bölücü değeri 16

2.2.2.2. PWM Sinyalinin Donanımsal Üretimi

Darbe genişlik modülasyonunda CCP1 veya CCP2 ucundan 10 bit çözünürlükte PWM çıkışı elde edilir. CCP1 ucu aynı zamanda PortC.2 olduğundan bu ucun çıkış yapılabilmesi için TrisC.2 biti lojik-0 yapılmalıdır. Aynı koşul CCP2 için de geçerlidir.

Bir PWM çıkışı iki büyüklük ile ifade edilir. Bunlardan birincisi taban zaman (periyot,Ton+Toff) diğeri de çıkışın lojik-1 değerinde kaldığı süre olan aktarma saykılıdır (Ton). PWM frekansı periyotun tersidir (1/T).

Şekil 2.13'te PWM modundaki sinyal şekli ve TMR2 ve PR2 yazmaçlarının bu sinyaldeki görevleri görülmektedir.



Şekil 2.13: PWM Modunda zamanlama sinyalleri

PWM Sinyalinin Periyotu

PWM periyotu PR2 yazmacı ile belirlenir. Periyot aşağıdaki formül ile hesaplanır;

Tpwm = [(PR2) + 1] • 4 • Tosc • (Tmr2 ön bölücü değeri)
Fpwm=1/Tpwm

Formülü ile de PWM frekansı hesaplanabilir. Tmr2 son skala değeri Pwm frekansının belirlenmesinde kullanılmaz.

Şekil 2.13'e göre TMR2 değeri PR2 değerine eşit olduğunda, bir sonraki artım saykılında aşağıdaki olaylar meydana gelir. TMR2 temizlenir.CCP1 biti lojik-1 olur. (PWM aktarma oranı %0 ise CCP1 lojik-0'da kalır). PWM aktarma saykılı CCPR1L'den CCPR1H'e aktarılır.

PWM Sinyalinin Aktarma Saykılı(Ton)

PWM Ton zamanı CCPR1L yazmacı ve CCP1CON yazmacının 5. ve 4. Bitlerine yazarak belirlenir. 10 bit çözünürlük ile bunu gerçekleştirmek mümkündür. CCPR1L yüksek 8 biti, CCP1CON ise düşük 2 biti saklar.

Ton =(CCPR1L:CCP1CON<5:4>) • TOSC • (TMR2 ön bölücü değeri)

Akarma oranı da daha önce ifade edildiği gibi;

D=Ton/T formülü ile bulunabilir.

CCPR1L ve CCPCON(5:4) bitlerine herhangi bir zaman diliminde yazılabilir. Fakat aktarma saykılı değeri, periyot tamamlanmadıkça diğer bir deyişle PR2 ile TMR2 arasında eşleşme meydana gelmedikçe CCPR1H'ye aktarılmaz. PWM modunda CCPR1H yazmacı sadece okunabilir özelliktedir.

Maksimum PWM çözünürlüğü aşağıdaki formül ile hesaplanır.

$$Çözünürlük = \frac{\log(\frac{Fosc}{Fpwm})}{\log(2)} \text{ bit}$$

Şekil 2. 14. Maksimum PWM Çözünürlüğü Hesaplama Formülü

Pwm aktarma saykılı periyottan uzun olursa CCP1 pini sıfırlanmayacaktır.

PWM Çalışmasının Kurulumu

CCP modülünü PWM moduna ayarlamak için aşağıdaki işlem basamakları gerçekleştirilir.

- PR2 yazmacı ile periyot değeri ayarlanır.
- CCPR1L ve CCP1CON(5:4) bitleri ile PWM aktarma saykılı (Ton süresi) ayarlanır.
- Trisc.2 biti sıfırlanarak CCP1 pini çıkış yapılır.
- TMR2 ön bölücü değeri ayarlanır ve T2CON yazmacı ile Timer2 aktif hale getirilir.
- CCP1CON modülünde PWM çalışması için çözünürlük ayarlanır.

PWM frekansı	**1,22 kHz**	**4,88 kHz**	**19,53 kHz**	**78,12 kHz**	**156,3 kHz**	**208,3 kHz**
Tmr2 Ön bölücü (1,4,16)	16	4	1	1	1	1
PR2 değeri	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Max çözünürlük	10	10	10	8	7	5,5

Şekil 2.15: Örnek-20 MHz'lik Osilatör Frekansında PWM Frekansları ve Çözünürlükler.

PIC16F877 mikrodenetleyicide PWM ve Timer2 çalışmasına etki eden diğer yazmaçlar Intcon, PIR1, PIR2, PIE1 ve PIE2'dir. Bu yazmaçlar PWM modülünün ve Timer2'nin çalışmasına etki ederler. Assembly dili ile PWM sinyali üretilecekse bu yazmaçlara da uygun bitler yazılmalıdır. Bu kısımda PIC-C hazır fonksiyonları kullanılacağından bu yazmaçlar hakkında bilgi verilmeyecektir. PIC-C fonksiyonlarının kullanımı ve bu fonksiyonlardaki hesaplamalar yukarıdaki hesaplamalarla bağlantılı bir şekilde işlenecektir.

2.2.2.3. PWM için Kullanılan PIC-C Komutları

PIC16F877 Mikrodenetleyicide PIC-C dili kullanarak PWM sinyali üretmek için 3 fonksiyonu kullanmak yeterlidir.

- **SET_PWM1_DUTY() [SET_PWM2_DUTY()]**

Aktarma saykılını(Ton) kurmak için 10 bitlik değer bu fonksiyon ile belirtilir. Düşük değerli bitler gerekli değilse 8 bitlik değerde bu fonksiyon ile kullanılabilir. Sadece PIC16F877 gibi CCP/PWM modülü içeren mikrodenetleyiciler için kullanılabilir. Herhangibir sürücü veya başlık dosyası gerektirmez.

Kullanımı: set_pwm1_duty(X)
set_pwm2_duty(X)

X değeri için 8 veya 16 bit genişliğinde bir sabit veya değişken kullanılabilir. Değer 10 bitlik bir sayı ise bu sayı için kullanılan değişken veya sabit en az 16 bitlik olmalıdır. Fonksiyonun dönüş değeri yoktur. Değer 8 bitlik bir sayı ise, 10 biti elde etmek için donanım tarafından lsb bitlerinde 2 adet lojik-0 biti ile kaydırma yapılır. 10 bitlik değer bir periyot içerisindeki PWM sinyalinin lojik-1'de kaldığı zaman miktarını saptamak için kullanılır. Ton zamanı (Aktarma saykılı) aşağıdaki formül ile bulunur.

$$Ton = X \frac{4}{Fosc} t2div$$

Şekil 2.16. Ton hesaplama formülü

Fosc: Kristal veya rezonatör ile belirlenen osilatör frekansıdır.

t2div: Timer2 ön bölücü değeridir. "setup_timer2()" fonksiyonu ile belirlenir. Bu fonksiyon hakkında ilerleyen kısımlarda bilgi verilmiştir.

- **SET_CCP1() [SET_CCP2()]**

CCP modülünün çalışma modunu ayarlar.

Kullanımı: setup_ccp1 (mod)

setup_ccp2 (mod)

"mod" bir sabittir. Her mikrodenetleyicinin başlık dosyasında geçerli sabitler tanımlanmıştır. Mod değeri için kullanılan sabit ifadeler PWM modu için aşağıda verilmiştir.

"CCP_OFF" : CCP modülünü pasif yapar
.
"CCP_PWM": Darbe genişlik modülatörünü (PWM) aktif yapar.

PWM modunda CCP modülü PWM sinyali üretir.

- **SETUP_TIMER_2()**

Timer2 yazmacını PWM sinyali için uygun değere ayarlar. Timer2 yazmacına sahip mikrodenetleyiciler için kullanılabilir.

Kullanımı: setup_timer_2 (mod, periyot, son bölücü)

Bu fonksiyonda mod, T2_DISABLED, T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16 ifadelerinden biridir. Mod osilatör saat darbesinin kaça bölüneceğini bildirir.

Periyot: Fonksiyon periyot katsayısıdır. Saat darbesi değerinin sıfırlandığı zamanı belirleyen 0-255 arasındaki int tamsayı tipinde bir değişkendir.

Son bölücü: Bir kesme sinyalinden önce zamanlayıcının kaç defa sıfırlanacağını belirleyen 1-16 arasında bir sayıdır. Örneğin bu sayı 1 olursa zamanlayıcı 1 kere, 2 olursa 2 kere resetlenir.

Aşağıda PWM sinyalinin hesaplanması ile ilgili bir örnek verilmiştir.

Örnek-2.2:

```
setup_ccp1(CCP_PWM); //CCP modülünü PWM için kurar.
setup_timer_2(T2_DIV_BY_4, 140, 0); //periyot 140, Osc 4'e böl, son bölücü 0
set_pwm1_duty(70); //%50 aktarma oranı, yukarıda periyot 140
//70 140'ın yarısı olduğu için D=%50 ayarlanır.
```

Aşağıdaki örnekte 996 Hz ve 2,475KHz frekanslarında iki ayrı ses Tonunun PWM modunda elde edildiği bir program örneği görülmektedir.

Örnek- 2.3:

```
void ses(sec)
{
setup_ccp1(CCP_PWM); //CCP1 modülünü PWM moduna ayarla
if(sec==2)
{
setup_timer_2(T2_DIV_BY_4, 100, 0);
set_pwm1_duty(50); // %50 aktarma oranı
delay_ms(1000); //1 saniye gecikme
// PWM sinyali üretilmeye devam eder
}
else
{
setup_timer_2(T2_DIV_BY_4, 250, 0);
set_pwm1_duty(125); // %50 aktarma oranı
delay_ms(500); //500 ms gecikme
}
```

set_pwm1_duty(0); //PWM sinyali kesilir. Çıkış sürekli lojik-0 olur.
} // Bu durumda ses Tonu elde edilemez.

Programda frekanslar
setup_timer_2(T2_DIV_BY_4, 100, 0);
fonksiyonu ile belirlenmiştir.
Aktarma oranı da aşağıdaki formül ile belirlenmiştir.
set_pwm1_duty(50);
Dikkat edilirse *set_pwm1_duty(50);* fonksiyonundaki 50 değeri s*etup_timer_2(T2_DIV_BY_4, 100, 0);* fonksiyonundaki 100 değerinin yarısıdır. Bu nedenle D oranı %50'dir.

Programda y değerine göre bir ses Tonu belirli bir süre üretilmiş ve sonra ses kesilmiştir. Ses kesilmiş olmasına rağmen 0 aktarma oranlı PWM sinyalinin üretilmesine devam edilir.

Aşağıdaki örnek PWM çalışıyorken ana programda bir ledin kontrol edildiği bir programdır. PWM çalışması PIC'i meşgul etmez. Ana programda PWM dışında başk çalışmalar gerçekleştirilebilir.

Örnek-2.4

```
//// Programda aktarma oranı program içinde belirleniyor. ////
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#define in input(PIN_D0)
#define out PIN_B0
byte port_b=6,port_d=8;
io_set()
{
set_tris_b(0x00);
set_tris_d(0xff);
}
pwmkur()
{
setup_ccp1(CCP_PWM); // CCP1 modülü PWM moduna kuruluyor.
setup_timer_2(T2_DIV_BY_16, 100, 1); //Periyot, mod, son bölücü ayarlanıyor.
}
pwm()
{
set_pwm1_duty(50); // %50 Aktarma
}
main()
{
io_set();
pwmkur(); // PWM kuruluyor.
pwm(); // CCP1 pininden PWM sinyali alınmaya başlar.
```

```
while(1)

if(in==0) output_high(out); //PD0'daki buTona basılırsa PB0'daki led ışık verir.
else output_low(out); //PWM çıkışı  bu çalışmadan etkilenmez.
}
}
```

PWM komutları aşağıda belirtildiği gibi çalışır.

1. setup_ccp1(CCP_PWM);

Bu komut ile CCP1 modülü PWM moduna ayarlanır.

setup_timer_2(T2_DIV_BY_16, 100, 1);

Bu komut satırında periyot 100, mod ise T2_DIV_BY_16'dır. 1 değeri son bölücü değeridir ve kesme sinyalinden önce timer2 1 kere sıfırlanır.

Bu fonksiyona göre PWM periyotu ve frekansı aşağıdaki formül ile hesaplanır.

T=(1/clock)*4*t2div*(period+1)

Programda kristal frekansı 20 MHz ve fonksiyon periyot katsayısı=100'dür. Buna göre;

Periyot (T);

(1/20000000)*4*16*101= 323,2 us olarak bulunur.

Frekans ise;

F=1/T=1/323,2us= 3,09 KHz bulunur.

set_pwm1_duty(50);

%50 aktarma oranlı PWM sinyalinin Ton süresini ayarlar. Bu süre;

$$Ton = X\frac{4}{Fosc}t2div$$

Şekil 2.17. Ton hesaplama formülü

formülü ile hesaplanır. Fonksiyonda X=50 ve t2div=16'dır. Buna göre;

Ton=50x4x(1/20000000)X16=160 µs bulunur.

$$\text{PWM Çözünürlüğü}=\frac{\log(\frac{Fosc}{Fpwm})}{\log(2)}\text{ bit'dir.}$$

Şekil 2.18. PWM Çözünürlüğü Hesaplama Formülü

Buna göre;

Pwm çözünürlüğünü "R" ile ifade edersek

$$R = \frac{\log(\frac{20x10^6}{3,09x10^3})}{\log(2)} = \frac{\log(6,47x10^3)}{0,3} = \frac{3,81}{0,3} = 12,7 \text{ bit}$$

Şekil 2.19. R değerinin hesaplanması

Örnek 2.5: CCP1 (PWM1) çıkışı ile 200 KHz kristal kullanarak pwm frekansı 50 Hz, duty cycle değeri 1 ila 2 ms arasında değiştirilebilen devre tasarlayınız. Duty değişkeninin alt ve üst değerlerini hesaplayınız. set_pwm1_duty(i); komutundaki duty değişkeni "i " 8 bit olarak seçilmiştir.

Verilenler

$f_{osc} = 200KHz$

$\mathbf{f_{PWM} = 50\ Hz}$

Istenenler

duty değişkeni_**min**= ?

duty değişkeni_**max**= ?

$$\mathbf{f_{komut} = \frac{f_{osc}}{4} = \frac{200KHZ}{4} = 50KHz}$$

TMR2 bölme oranı 1, 4, 16 değerlerini alabilir. Biz 4 değerini seçtik.

$$\mathbf{T_{PWM} = T_{komut} x (PR2 + 1) x (TMR2\ bölme\ oranı)}$$

$$\mathbf{0,02 = 2 * 10^{-5} * (PR2 + 1) * 4 \rightarrow PR2 = 249\ bulunur}$$

PR2 değeri 0 ila 255 arasındadır. PR2= 249 değeri uygundur. Duty değişkenleri PR2 değerinden küçük olmalıdır . Aksi halde PWM sinyalinin fazı, frekansı, periyodu gibi değerler değişir. PR2 çok küçük seçilirse duty değişkenleri çok küçük bir alana sıkışacak, değişken sayısı azalacak, adım açısı büyüyecektir. PR2 değerini uygun hale getirmek için farklı frekanslı osilatör veya farklı TMR2 bölme oranları seçilebilir.

$$\mathbf{f_{PWM} = \frac{1}{T_{PWM}} = \frac{1}{0,02} = 50\ Hz}$$

Şekil 2.20. fPWM değerinin hesaplanması

set_pwm1_duty(i); komutunda kullanılacak kullanılacak duty değişkeni "i "nin alt ve üst değerlerinin hesaplanması gerekmektedir. i değişkeni 8 veya 10 bit seçilebilir. i değişkeni 8 bit olarak seçilirse derleyici yazılan bu değeri 2 kez sola kaydırır ve 10 bitlik değere çevirir. 2 kez sola kardırma işlemi ile sayı değeri 4 ile çarpılmış olur ve aşağıdaki formül kullanılır.

Duty değişkeninin minumum değerini hesaplayalım.

$$T_{\mathbf{PWM_{DUTY_{min}}}} = duty\ değişkeni_{min} * \mathbf{T_{OSC}} * \mathbf{(TMR2\ bölme\ oranı) * 4}$$

değişken tipini tam sayı seçtiğimiz için 12,5 değerini aşağı yada yukarı yuvarlayabiliriz. Biz 12 olarak seçtik.

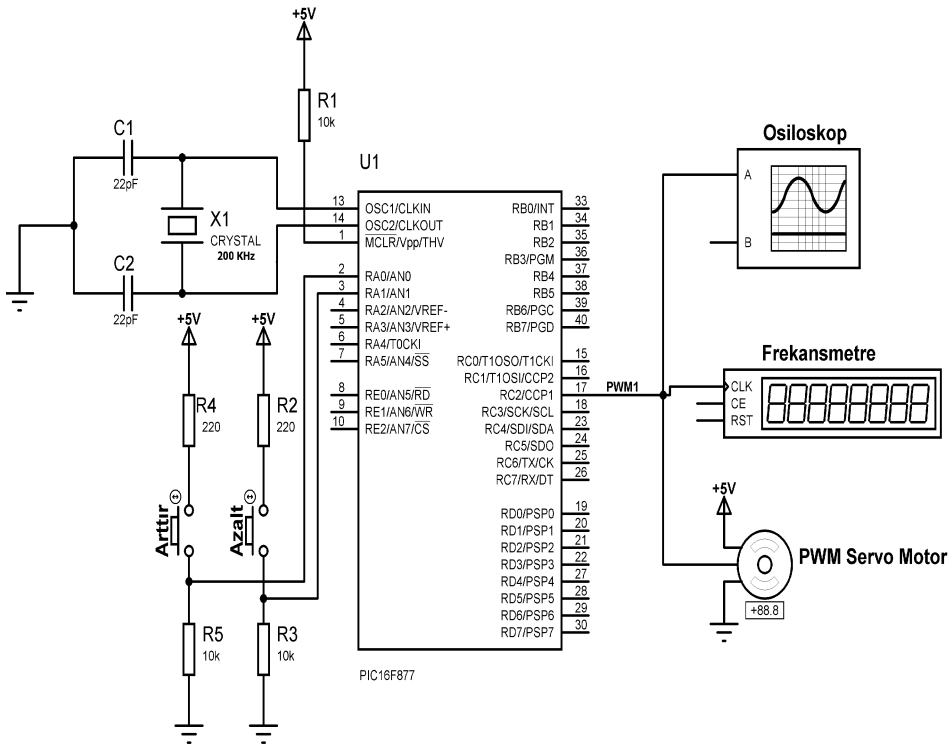
Duty değişkeninin maksimum değerini hesaplayalım.

$$T_{\mathbf{PWM_{DUTY_{max}}}} = duty\ değişkeni_{max} * \mathbf{T_{OSC}} * \mathbf{(TMR2\ bölme\ oranı) * 4}$$

Duty değikeni 12 iken servo motor -90 derecede, Duty değişkeni 25 iken servo motor +90 dereceye gitmektedir. Duty değişkeninin bir sayı artması yada azalması yaklaşık 14 derecelik açısal harekete neden olmaktadır. Başka bir deyişle servo motorumuz 14 derece civarında açı adımları ile kontrol edilebilmektedir.

$$\mathbf{Adım\ açısı} = \frac{toplam\ açısal\ değişim}{duty\ değişkeni_{max} - duty\ değişkeni_{min}} = \frac{\mathbf{180}}{\mathbf{25-12}} \cong \mathbf{14°}$$

Şekil 2.21. Adım açısının hesaplanması



Şekil 2.22 : Örnek 2.5 in bağlantı şeması

Örnek 2.5 in PIC C kodu:

```
/*********************************************************
    Donanımsal PWM ile servo motor kontrolu
*********************************************************/
#include <16f877.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.

// Konfigürasyon ayarları
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
NODEBUG, NOCPD
#use delay (clock=200000) // Osilatör frekansı 200KHz belirtiliyor.

#use fast_io(a) //Port yönlendirme komutları A portu için geçerli
#use fast_io(c) //Port yönlendirme komutları C portu için geçerli

int i=12; // Tamsayı tipinde değişken tanımlanıyor

/********* ANA PROGRAM FONKSİYONU********/

void main ( )
{
setup_psp(PSP_DISABLED);        // PSP birimi devre dışı
setup_timer_1(T1_DISABLED);     // T1 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);    // ANALOG giriş yok
setup_adc(ADC_OFF);             // ADC birimi devre dışı

set_tris_a(0x03);   // RA0 ve RA1 pinleri giriş
set_tris_c(0x00);   // RC0, RC1 ve RC2 pini çıkış

setup_ccp1(CCP_PWM);  // CCP1 birimi PWM çıkışı olarak ayarlandı

setup_timer_2(T2_DIV_BY_4,249,1); // Timer2 ayarları yapılıyor
set_pwm1_duty(i); // PWM1 çıkışı görev çevrimi(duty cycle) belirleniyor.
while(1) // Sonsuz döngü
{
if (input(pin_a0)) // Eğer RA0 girişine bağlı butona basılırsa
{
delay_ms(20);   // Buton arkı önleme gecikmesi
while(input(pin_a0)); // Basılı butondan el çekile kadar bekle
i+=1; // i=i+1 anlamındadır.
if (i>25) // i değeri 25'dan büyükse i değeri 25 olsun
i=25;
set_pwm1_duty(i); // PWM1 çıkışı görev saykılı belirleniyor
}

if (input(pin_a1)) // Eğer RA1 girişine bağlı butona basılırsa
{
```

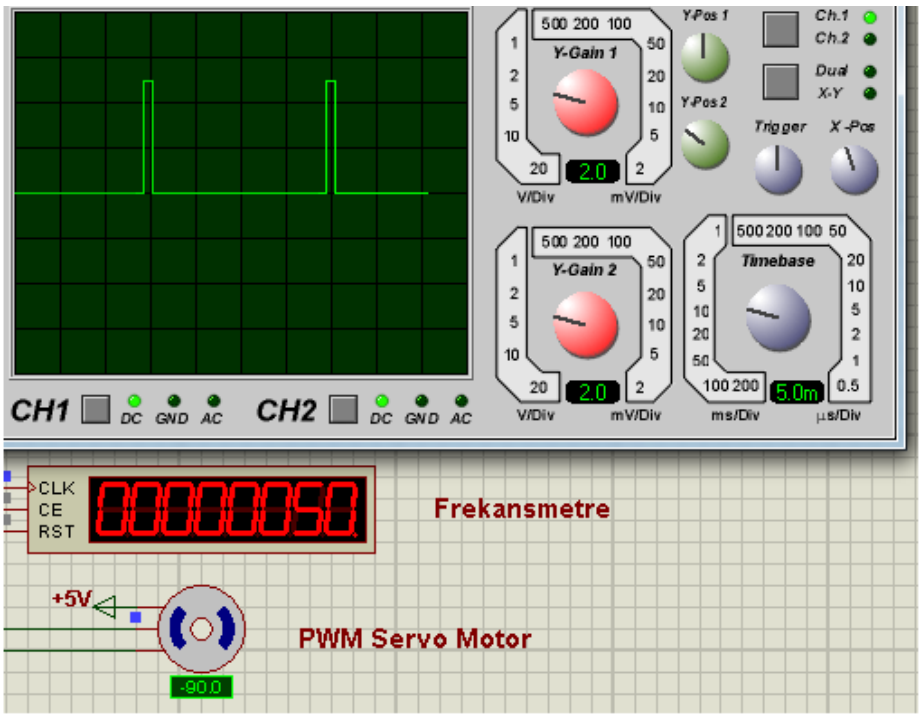
```
delay_ms(20);  // Buton arkı önleme gecikmesi
while(input(pin_a1)); // Basılı butondan el çekile kadar bekle
i-=1;  // i=i-1 anlamındadır.
if (i<12) // i değeri 12'den küçük ise i değeri 12 olsun
i=12;
set_pwm1_duty(i); // PWM1 çıkışı görev saykılı belirleniyor
}
}
}
```

AÇIKLAMA:

#use delay (clock=200000) komutu ile gecikme fonksiyonu için kullanılacak osilatör frekansı 200KHz olarak belirtiliyor. **setup_ccp1(CCP_PWM);** komutu CCP1 biriminin PWM amaçlı kullanılacağı belirtliyor. **setup_timer_2(T2_DIV_BY_4,249,1);** komutu ile Timer2 ayarları yapılıyor. **set_pwm1_duty(i);** komutu ile PWM1 çıkışı görev çevrimi(duty cycle) belirleniyor . Buradaki i karakteri duty değişkenini ifade etmektedir. duty değişkeninin ilk değeri 12 dir. **if (input(pin_a0))** komutu ile RA0 girişine bağlı butona basılıp basılmadığı kontrol edilmektedir. **delay_ms(20);** komutu ile buton arkları sönene dek 20ms süre için denetleyiciye A0 pininden sinyal girişi ertelenmiştir. while(input(pin_a1)); komutu ile kullanıcı butondan elini çekene kadar başka işlem yapılmaması sağlanmıştır. **i+=1**; komutu i=i+1 ifadesi ile aynıdır. Buradaki "i" duty değişkenini ifade etmektedir. RA0 pinine bağlı butona her basışta i değişkeninin içeriği bir sayı artmaktadır. if (i>25) komutu ile "i " değeri 25' ten büyük olup olmadığı kontrol edilir. **i=25;** komutu ile **i** değerinin alabileceği üst değer 25 olarak sabitlenir. **set_pwm1_duty(i);** komutu ile pwm sinyalinin yeni görev çevrimi belirleniyor. **if (input(pin_a1))** komutu ile RA1 girişine bağlı butona basılıp basılmadığı kontrol edilmektedir. i-=1; komutu i=i-1 ifadesi ile aynı anlamda olup butona her basışta "i" değişkeninin içeriğini 1 azaltmaktadır. if (i<12); ve i=12; komutları ile "i" değişkenin alt değeri 12'ye sabitleniyor. **set_pwm1_duty(i);** komutu ile pwm sinyalinin yeni görev çevrimi belirleniyor.

Örnek 2.5 in Isıs Programında Testi:

Simülasyonun ilk açılışlında servo motor -90° konumuna gelir. PWM sinyalinin görev çevrimi(duty cycle) osiloskop ekranından ve frekans değeri frekansmetreden Şekil 2.16 da ki gibi görülür.



Şekil 2.23: Örnek 2.5 in Isıs Programında Testi

Arttır butonuna her basışta açı değeri -90° değerinden +90° değerine doğru 14,4° lik adımlarla artar. Arttır butonuna 1. basışta motor -82,8° 'ye, 2. basışta 68,4° 'ye, 3. basışta 54° 'ye, 4. basışta 39,6'ye konumlandı. Dikkat edilirse ilk basışta 7,2°'lik bir açısal değişim, diğerlerinde ise eşit 14,4° lik açısal değişim olmuştur. Duty değişkeni "i" nin ilk değeri 12 olup, bu değer için görev T_{on} süresi 0,96 ms dir. -90° deki hareket kısıtlamasından dolayı $T_{(on)}$ süresinin 0,96 ms olması motorun -97,2 ° lik açı değerine gelmesini sağlamaz. Arttır butonuna diğer basışlarda 14,4° lik düzenli değişimlerin olduğu gözlenir. Azalt butonuna her basışta ise açı değerinin +90° değerinden -90° değerine doğru 14,4° lik adımlarla azaldığı izlenir. Osiloskop ekranından T_{on} süresinin 1-2 ms arasında değişimi gözlenebilir. Laboratuvar ortamında yapılacak gerçek uygulamalarda RC servo motorun adım açılarının değişiminin çok düzenli ve hasas olması beklenmemelidir. Pratikte RC servo motor hassas pozisyonlama için kullanılmaz.

UYGULAMA FAALİYETİ

CCP1 (PWM1) çıkışı ile 200 KHz kristal kullanarak pwm frekansı 50 Hz, duty cycle değeri 1 ila 2 ms arasında değiştirilebilen devre tasarlayınız. Duty değişkeninin alt ve üst değerlerini hesaplayınız. set_pwm1_duty(i); komutunda kullanılacak duty değişkeni "i" 16 bit olarak seçilecektir. Gerekli hesaplamaları yaparak PIC C kodunu yazınız. Devreyi uygulayınız. Bağlantı şeması için şekil 2.15 i kullanınız.

İşlem Basamakları	Öneriler
➢ Şekildeki devre için gereken malzemeleri ve araç gereci hazırlayınız.	➢ 5V gerilim kaynağı kaliteli bir kaynak olmalıdır.
➢ PIC16F877 mikrodenetleyicinin ve kullandığınız servo motorun katalog bilgilerini inceleyiniz.	➢ Çalışmalarını ve karakteristik özelliklerini inceleyiniz.
➢ Duty değişkeninin alt veüst değerlerini, kaç derecelik adım açıları ile hareket edeceğini hesaplayınız.	➢ Modülünüzdeki Örnek 2,5'ten faydalanabilirsiniz.
➢ Hesapladığınız değerleri kullanarak PIC C kodunu yazınız.	➢ Örnekteki PIC C kodunda int i komutu yerine int16 i ifadesini yazınız. Duty değişkeninin alt veüst değerlerini programınızda ilgili yerlere yazınız.
➢ Yazdığınız programı derleyeyerek hex dosyasını oluşturunuz ve ISIS programında test ediniz. Servo motor kaç derecelik adım açıları ile hareket ettiğini gözleyiniz.	➢ ISIS programında arttır veya azalt butonuna basarak adım açılarını gözleyiniz. Adımlar arasındaki farkı hesaplapyarak adım açısını hesaplayınız.
➢ Oluşturulan hex dosyayı denetleyiciye yükleyiniz, denetleyiciyi bord üzerine takarak öğretmeniniz gözetiminde devreyi test ediniz.	➢ Denetleyiciyi borda takıp çıkarırken pinlerin yamulup kırılmamasına özen gösteriniz.
➢ -90, 0, +90 derecelerdeki Ton süresini osiloskoptan ölçünüz. Ton süreleri hesaplanandan farklı ise nedenini açıklayınız.	➢ Ton sürelerini hassas bir şekilde ölçmek için sinyalin lojik 1 olduğu kısmın yatayda ekranı kaplaması için osiloskopun time/div anahtarı ile ayar yapınız.

ÖLÇME VE DEĞERLENDİRME

Aşağıda soruları dikkatlice okuyarak doğru seçeneği işaretleyeniz

1. İstenilen bir açıya doğrudan konumlanabilen motor türü aşağıdakilerden hangisidir?
A) Step Motor
B) DC motor
C) AC motor
D) Servo motor

2. Sahip olduğu bir mil üzerinden gücünü dışarıya aktarabilen elemana ne ad verilir?
A) Kam
B) Servo
C) Çark
D) Şaft

3. Servo motora uygulanan sinyalin Ton süresinin değişmesi sonuç olarak aşağıdakilerden hangisini doğrudan değiştirir?
A) Açı
B) Hız
C) İvme
D) Tork

4. Servo motor iç yapısında tork değerini arttıran yapıya ne ad verilir?
A) Mıknatıs
B) Mil
C) Dişli kutusu
D) Rotor

5. Servo motorun dönme açısı hangi eleman ile algılanır?
A) Kumanda devresi
B) Geri besleme potansiyometresi
C) Motor mili
D) Strator

6. Aşağıdakilerden hangisi motor milinin istenenen açı değerine ayarlar?
A) Rotor
B) Geri besleme potansiyometresi
C) Motor bobini
D) Kumanda devresi

7. Kontrol sinyali Ton süresi 1,25 ms ise motor milinin katedeceği açı teorik olarak aşağıdakilerden hangisidir?
A) -30
B) -45
C) +30
D) +45

8. +60 derecelik açı hareketi için Ton süresi teorik olarak ne olmalıdır?
A) 1,5 ms
B) 1,6 ms
C) 1,8 ms
D) 2 ms

9. Periyot sabit kalmak şartıyla Ton süresinin değişmesi ile elde edilen sinyal modülasyon tekniğine ne ad verilir?
A) FM
B) AM
C) PCM
D) PWM

10. Bir servo motoru sürmek için tasarlanan elektronik devrenin güç kısmına ne ad verilir?
A) Kumanda devresi
B) Güç kaynağı
C) Servo dişlisi
D) Servo sürücüsü

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıda soruları dikkatlice okuyarak doğru seçeneği işaretleyeniz

1. PIC16F877 mikrodenetleyicinin veri belleği kaç B'dır?
A) 8
B) 38
C) 68
D) 368

2. PIC16F877 mikrodenetleyicide PWM sinyali hangi porttan veya portlardan elde edilebilir?
A) PortA
B) PortB
C) PortC
D) Hepsi

3. PIC16F877'de kesme kontrolü için hangi yazmaç kullanılır?
A) Intcon
B) PR2
C) PIR2
D) Status

4. LCD'de ekrana yazılacak olan veri aşağıdaki birimlerden hangisi içerisinde kayıtlıdır?
A) CGRAM
B) DDRAM
C) CPU
D) CGROM

5. LCD ekranında istenilen koordinata bir karakter veya karakter dizisini yazan komut aşağıdakilerden hangisidir?
A) Lcd_putc()
B) Lcd_kur()
C) Lcd_init()
D) Dsply_yaz()

6. Aşağıdaki motor türlerinden hangisinde açı geri beslemesi bir potansiyometre ile yapılır?
A) Step motor
B) DC motor
C) AC motor
D) Servo motor

7. Aşağıdakilerden hangisi servomotorda tork değerini artırır?
A) Kam
B) Potansiyometre
C) Çark Grubu
D) Mil

8. Kontrol sinyali Ton süresi 1,75 ms ise motor milinin katedeceği açı teorik olarak aşağıdakilerden hangisidir?
A) -30
B) -45
C) +45
D) +30

9. Aşağıdaki modüllerden hangisinin PWM sinyali üzerinde bir etkisi yoktur?
A) Timer1
B) CCP1
C) TRISC
D) Timer2

10. PWM modülünde periyot katsayısı değerini 50, t2div değerini 4 ve son bölücü değerini 1 yapan komut aşağıdakilerden hangisidir?
A) setup_timer_2(T2_DIV_BY_50,1,4)
B) setup_timer_2(T2_DIV_BY_4,1,50)
C) setup_timer_2(T2_DIV_BY_4,50,1)
D) setup_timer_2(T2_DIV_BY_1,4,50)

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Uygulamalı Test"e geçiniz.

KONTROL LİSTESİ

Bu faaliyet kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet**, kazanamadığınız becerileri **Hayır** kutucuğuna (**X**) işareti koyarak kendinizi değerlendiriniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. PIC16F877 mikrodenetleyiciyi istenen sisteme uygun olarak kullanabildin mi?		
2. LCD-Mikrodenetleyici bağlantısını doğru olarak yapabildin mi?		
3. LCD'yi istenen amaca uygun olarak hazır sürücü dosyası ile programlayabildin mi?		
4. LCD'yi istenen amaca uygun olarak sürücü dosyası oluşturup programlayabildin mi?		
5. PWM sinyalini yazılım ile elde edebildin mi?		
6. PWM sinyalini donanım özellikleri ile elde edebildin mi?		
7. PIC16F877-Servo motor bağlantısını yapabildin mi?		
8. Servo motoru istenen açı değerine getirebildin mi?		

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALIYETİ -1'DEKİ UYGULAMA CEVABI

```
//******* Öğrenme Faaliyeti-1  LCD Sürücü Dosyası ******************
//********************* uyg1_LCDriver.c  ****************************

#define rs  pin_b0 // LCD'nin RS ucu RB0 pinine bağlı
#define e   pin_b1 // LCD'nin E   ucu RB1  pinine bağlı

//******LCD'ye Komut Gönderme Fonksiyonu **********
void lcd_komut(byte komut)
{
output_c(komut>>4);  // Komutun yüksek değerli 4 bitini gönder
output_low(rs);         //  LCD komut almak için ayarlandı
delay_cycles(1);        //  1 komut saykılı bekle
output_high(e);         //  E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);        //  1 komut saykılı bekle
output_low(e);          //   E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);           //   5 msn gecikme veriliyor
output_c(komut&0x0F); // Komutun düşük değerli 4 bitini gönder
output_low(rs);              // LCD veri almak için ayarlandı
delay_cycles(1);             // 1 komut saykılı bekle
output_high(e);              // E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);             // 1 komut saykılı bekle
output_low(e);        // E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);          // 5 msn gecikme veriliyor
}

//******* LCD'ye Veri Gönderme Fonksiyonu **********
void lcd_veri(byte veri)
{
output_c(veri>>4);    // Verinin yüksek değerli 4 bitini gönder
output_high(rs);      //   LCD veri almak için ayarlandı
delay_cycles(1);      //  1 komut saykılı bekle
output_high(e);       //   E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);      //  1 komut saykılı bekle
output_low(e);        //   E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
delay_ms(5);          //   5 msn gecikme veriliyor

output_c(veri&0x0F); // Verinin düşük değerli 4 bitini gönder
output_high(rs);      //     LCD veri almak için ayarlandı
delay_cycles(1);      //    1 komut saykılı bekle
output_high(e);       //     E ucu ilk önce lojik-1 yapılıyor sonra lojik-0'a çekilecek
delay_cycles(1);      //     1 komut saykılı bekle
output_low(e);        //     E ucu lojik-0 yaplıp komut uygula yetkisi veriliyor
```

```
delay_ms(5);        //      5 msn gecikme veriliyor
}

//******* LCD'de İmlec Konumlandırma Fonksiyonu ********
void imlec(byte satir, byte sutun)
{
f (satir==1)      // Eğer 1. satır seçili ise
lcd_komut(0x80|(sutun-1));

if (satir==2)      // Eğer 2. satır seçili ise
lcd_komut(0xC0|(sutun-1));
}

//********* LCD Başlangıç Ayarları Fonksiyonu ******
void lcd_hazirla()
{
int i=0;
output_low(rs); // RS ucu lojik-0
output_low(e);  // E ucu lojik-0
delay_ms(100);   // LCD enerjlendiğinde LCD'nin hazır olması için beklenen süre

for(i=0;i<3;i++)    // LCD'ye 3 kez 0x03 fonksiyon komutu gönderiliyor
{
lcd_komut(0x03);
delay_ms(10);      // 10 msn gecikme veriliyor
}
lcd_komut(0x02); // LCD'ye 4 bit iletişim komutu gönderiliyor
lcd_komut(0x28); // 4 bit iletişim, 2 satır, 5x8 dot matris seçildi
lcd_komut(0x08); // Display Kapalı, imleç gösterilmez ve yanıp sönmez
lcd_komut(0x01); // Display sil. İmleç 1.satır 1.sütunda
lcd_komut(0x06); // Her gönderilen karakterden sonra imleç bir sağa gitsin
lcd_komut(0x0C); // Display açık,imleç gösterilmez ve yanıp sönmez.
}

/*********************************************************
Öğrenme Faaliyeti-1  Uyg_1.c  Dosyası ile LCD Uygulaması

*********************************************************/

#include <16f877.h>     // Dnetleyicinin başlık dosyası tanıtılıyor.

#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
NODEBUG, NOCPD // Denetleyicinin konfigürasyon ayarları

#use delay (clock=4000000) // Kullanılacak osilatör frekansı belirtiliyor.
```

```
#include < uyg1_LCDriver.c > // sürücü  dosyası programa tanıtılıyor

/********************** ANA PROGRAM **********************/

void main ( )
{
setup_psp(PSP_DISABLED);          // PSP birimi devre dışı
setup_timer_1(T1_DISABLED);       // T1 zamanlayıcısı devre dışı
setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);   // ANALOG giriş yok
setup_adc(ADC_OFF);                // ADC birimi devre dışı
setup_CCP1(CCP_OFF);               // CCP1 birimi devre dışı
setup_CCP2(CCP_OFF);               // CCP2 birimi devre dışı

lcd_hazirla(); // LCD nin ilk açılış ayarları yapılıyor

while(1) // Sonsuz döngü
{
imlec(1,1);                                    // İmlec 1.satır 3.sütunda
printf(lcd_veri,"PARALEL LCD");          // LCD'ye veri gönderiliyor
delay_ms(1000);                               // 1 sn gecikme veriliyor
lcd_komut(0x01);                               // ekran siliniyor.
imlec(2,1);                          // İmlec 2.satır 1.sütuna konumlandırılıyor.
printf(lcd_veri,"PARALEL LCD");// LCD'ye veri gönderiliyor.
delay_ms(1000);                               // 1 sn gecikme veriliyor
lcd_komut(0x01);                               // ekran siliniyor.
}
}
```

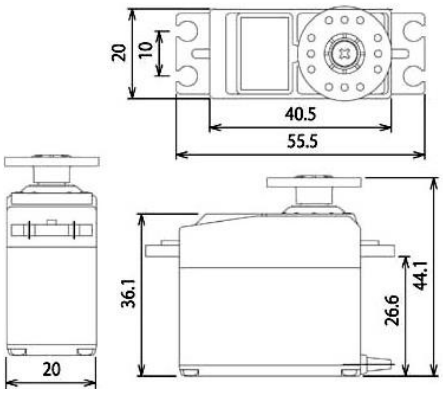
ÖĞRENME FAALIYETİ -1'İN CEVAP ANAHTARI

1	**A**
2	**C**
3	**C**
4	**C**
5	**D**
6	**C**
7	**D**
8	**D**
9	**C**
10	**B**

ÖĞRENME FAALIYETİ -2 ‘DEKİ UYGULAMA CEVABI

Servo Motor Katalog Bilgiler
Stok Numarası: FUTM0031
Üretici Numarası: S3003

Kontrol Sistemi: PWM sinyali ile Ton= 1520 µsn için Nötr pozisyonu (0°)
Gerekli Pals: Tepeden tepeye 3-5 Volt kare dalga
Çalışma Gerilimi: 4,8-6,0 Volt
Çalışma sıcaklığı: -20 ile +60 derece arası
Hareket Hızı (4.8V): 0.23 saniye/60° (yüksüz durumda 4,8 V luk çalışma geriliminde 60° derecelik açısal hareketi 0,23 sn de gerçekleştirir.)
Hareket Hızı (6.0V): 0.19 saniye /60° (yüksüz durumda 6 V luk çalışma geriliminde 60° derecelik açısal hareketi 0,19 sn de gerçekleştirir.)
Maksimum Tork (4.8V): 3.2kg-cm (4,8 V luk çalışma geriliminde 1cm uzunluğundaki çubuğa bağlanan 3.2kg yükü hareket ettirir)
Maksimum Tork (6.0V): 4.1kg-cm (6 V luk çalışma geriliminde 1cm uzunluğundaki çubuğa bağlanan 4,1 kg yükü hareket ettirir)
Çalışma açısı: Tek taraf 45 Derece için Ton ayar süresi 400 µsn
360 dereceye modifiyesi: evet mümkün
Yön: 1520-1900 µsn palsler ile saat tersi yönde hareket eder
Çektiği Akım (4.8V): 7.2mA
Çektiği Akım (6.0V): 8mA
Motor Tipi: 3 Kutuplu Ferit
Potansimetre sürme: dolaylı sürme
Rulman tipi: Plastik
Dişli tipi: Naylon
Bağlantı kablosu uzunluğu: 30cm
Boyutlar: 41 x 20 x 36mm
Ağırlık: 37.2g



Servo motorun 0, +90, -90, +max ve-max açıları için gereken Ton sinyalleri:
yapılan denemelere göre;
0 derece için 1,29 ms
- 90 derece için 0,31 ms
+90 derece için 2,25 ms
-max için 0,2 ms
+max için 2,33 ms bulunmuştur.

Servo motor milinin hareket aralığı 180 dereceden büyükür. Tahminen 200 derece civardır. Ölçülen değerler katalog değerlerinden farklıdır. Örneğin nötr pozisyonu (0 derece) için katalogta belirtilen Ton değeri 1,52 ms ölçülen ise 1,29 ms. Bu farklılık servonun analog servo olması ve hasiyetinin düşük olmasından ve/veya imalat hatası ve/veya insan kaynaklı ölçme hatasından kaynaklanabilir.

Programda kullanılan Duty değişkenin tipi 8 bit değilde 16 bit olduğundan aşağıdaki formül kullanılır.

$$duty\ değişkeni_{min} = 50$$

$$duty\ değişkeni_{max} = 100$$

$$\textbf{Adım açısı} = \frac{180}{100 - 50} = 3,6°$$ olacaktır. Görüldüğü gibi duty değişkenini 8 bit yerine 16 bit seçince adım açısı küçüldü. Daha hassas bir kontrol sağlandı. Örnek 2.5 teki programda bazı küçük değişiklikler yapılmalıdır. int i ifadesi int16 i şeklinde değiştirilmeli, duty değişkeninin alt ve üst değerleri ilgili yerlere yazılmalıdır.

```
/*********************************************************
Uygulama 2: Donanımsal PWM ile servo motor kontrolu
*********************************************************/
#include <16f877.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.

// Konfigürasyon ayarları
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,
NODEBUG,NOCPD

#use delay (clock=200000) // Osilatör frekansı 200KHz belirtiliyor.

#use fast_io(a) //Port yönlendirme komutları A portu için geçerli
#use fast_io(c) //Port yönlendirme komutları C portu için geçerli

int16 i=50; // Tamsayı tipinde değişken tanımlanıyor

/********* ANA PROGRAM FONKSİYONU********/

void main ( )
{
setup_psp(PSP_DISABLED);        // PSP birimi devre dışı
setup_timer_1(T1_DISABLED);     // T1 zamanlayıcısı devre dışı
setup_adc_ports(NO_ANALOGS);    // ANALOG giriş yok
setup_adc(ADC_OFF);             // ADC birimi devre dışı

set_tris_a(0x03);   // RA0 ve RA1 pinleri giriş
set_tris_c(0x00);   // RC0, RC1 ve RC2 pini çıkış
setup_ccp1(CCP_PWM);        // CCP1 birimi PWM çıkışı olarak ayarlandı
setup_timer_2(T2_DIV_BY_4,249,1); // Timer2 ayarları yapılıyor
set_pwm1_duty(i); // PWM1 çıkışı görev çevrimi(duty cycle) belirleniyor.
while(1) // Sonsuz döngü
{
if (input(pin_a0)) // Eğer RA0 girişine bağlı butona basılırsa
{
delay_ms(20);   // Buton arkı önleme gecikmesi
while(input(pin_a0)); // Basılı butondan el çekile kadar bekle
i+=1; // i=i+1 anlamındadır.
if (i>100) // i değeri 25'dan büyükse i değeri 25 olsun
i=100;
set_pwm1_duty(i); // PWM1 çıkışı görev saykılı belirleniyor
}

if (input(pin_a1)) // Eğer RA1 girişine bağlı butona basılırsa
{
delay_ms(20);  // Buton arkı önleme gecikmesi
while(input(pin_a1)); // Basılı butondan el çekile kadar bekle
i-=1;  // i=i-1 anlamındadır.
```

```
if (i<50) // i değeri 12'den küçük ise i değeri 12 olsun
i=50;
set_pwm1_duty(i); // PWM1 çıkışı görev saykılı belirleniyor
}
}
}
```

ÖĞRENME FAALIYETİ -2'NİN CEVAP ANAHTARI

1	**D**
2	**B**
3	**A**
4	**C**
5	**B**
6	**D**
7	**B**
8	**C**
9	**D**
10	**D**

MODÜL DEĞERLENDİRME CEVAP ANAHTARI

1	**D**
2	**D**
3	**A**
4	**B**
5	**A**
6	**D**
7	**C**
8	**C**
9	**A**
10	**C**

KAYNAKÇA

- Megep "Endüstriyel Otomasyon Teknolojileri Alanı" **"Mikrodenetleyici Uygulamaları-1 Modülü"** Ankara 2007
- Çiçek, Serdar **"CCS C ile PIC Programlama"** ALTAŞ YAYINCILIK, KASIM 2012